



WebSphere Application Server Liberty Core 8.5.5

Version 8 Release 5

Contents

Chapter 1. WebSphere Application

Server Liberty Core: Overview 1

Architecture	1
Java EE 7 programming model support	4
Java EE 6 programming model support	8
Supported Java EE 6 and 7 feature combinations	11
Java EE 7 behavior changes	12
Enterprise OSGi programming model support	13
Liberty externals support	14
Server configuration	16
Microsoft Active Directory LDAP Filters	
(activatedLdapFilterProperties)	17
Administrator Role (administrator-role)	17
API Discovery (apiDiscovery)	18
Application (application)	18
Application Manager (applicationManager)	31
Application Monitoring (applicationMonitor)	32
Authentication Cache (authCache)	33
Authentication Data (authData)	33
Authentication Filter (authFilter)	33
Authentication (authentication)	36
Feature Authorization Role Mapping	
(authorization-roles)	36
Basic User Registry (basicRegistry)	37
BELL (bell)	38
OSGi Applications Bundle Repository	
(bundleRepository)	40
Contexts And Dependency Injection (CDI) V1.2	
(cdi12)	41
CDI Container (cdiContainer)	41
Channel Framework (channelfw)	42
Classloading (classloading)	42
Certificate Authority Signed Certificate	
(collectiveCertificate)	43
Collective Member (collectiveMember)	43
Configuration Management (config)	44
Connection Manager (connectionManager)	47
Constrained Delegation (constrainedDelegation)	49
Thread Context Propagation (contextService)	49
Cross-Origin Resource Sharing (cors)	53
Custom LDAP Filters	
(customLdapFilterProperties)	54
Data Source (dataSource)	54
Distributed Map (distributedMap)	104
IBM Lotus Domino LDAP Filters	
(domino50LdapFilterProperties)	108
Novell eDirectory LDAP Filters	
(edirectoryLdapFilterProperties)	108
EJB Application (ejbApplication)	108
EJB Container (ejbContainer)	115
Enterprise Application (enterpriseApplication)	129
Event Logging (eventLogging)	142
Executor Management (executor)	143
Feature Manager (featureManager)	146
User Registry Federation (federatedRepository)	146

Fileset (fileset)	156
Host Authentication Information (hostAuthInfo)	157
Host Singleton (hostSingleton)	159
HTTP Access Logging (httpAccessLogging)	159
HTTP Dispatcher (httpDispatcher)	160
HTTP Transport Encoding (httpEncoding)	160
HTTP Endpoint (httpEndpoint)	163
HTTP Options (httpOptions)	167
HTTP Proxy Redirect (httpProxyRedirect)	169
HTTP Session (httpSession)	169
HTTP Session Database (httpSessionDatabase)	175
HTTP Whiteboard (httpWhiteboard)	181
IBM Tivoli Directory Server LDAP Filters	
(idsLdapFilterProperties)	182
Include (include)	182
Sun Java System Directory Server LDAP Filters	
(ipplanetLdapFilterProperties)	183
JAAS Login Context Entry	
(jaasLoginContextEntry)	184
JAAS Login Module (jaasLoginModule)	184
Java 2 Security (javaPermission)	187
JDBC Driver (jdbcDriver)	188
JNDI Entry (jndiEntry)	190
JNDI Object Factory (jndiObjectFactory)	190
JNDI Reference Entry (jndiReferenceEntry)	192
JNDI URL Entry (jndiURLEntry)	195
JPA Container (jpa)	195
JSP Engine (jspEngine)	196
Keystore (keyStore)	197
LDAP User Registry (ldapRegistry)	199
Shared Library (library)	211
Logging (logging)	214
Logstash Collector (logstashCollector)	216
LTPA Token (ltpa)	217
Mail Session Object (mailSession)	217
Managed Executor (managedExecutorService)	218
Managed Scheduled Executor	
(managedScheduledExecutorService)	222
Managed Thread Factory	
(managedThreadFactory)	227
Default Mime Types (mimeTypes)	232
Monitor (monitor)	232
Netscape Directory Server LDAP Filters	
(netscapeLdapFilterProperties)	233
OAuth Role Map (oauth-roles)	233
OAuth Provider Definition (oauthProvider)	236
OpenId Authentication (openId)	294
OpenID Connect Client (openidConnectClient)	298
OpenID Connect Server Provider	
(openidConnectProvider)	304
OSGi Application (osgiApplication)	368
OSGi Applications (osgiApplications)	370
OSGi Library (osgiLibrary)	370
Web Server Plugin (pluginConfiguration)	372
Quick Start Security (quickStartSecurity)	380
Remote File Access (remoteFileAccess)	380

Request Timing (requestTiming)	381	Video: Thoughts on Liberty: Interview with Alasdair Nottingham	668
SAML Web SSO 2.0 Authentication (samlWebSso20)	381	Video: Touring Admin Center	669
IBM SecureWay Directory Server LDAP Filters (securewayLdapFilterProperties)	392	Video: Using the IBM WebSphere Liberty Repository to enhance Liberty environments	670
Spnego Authentication (spnego)	392	Video: Why Liberty? Performance that scales	672
SSL Repertoire (ssl)	396	Video: Why Liberty? Fast application development	675
SSL Default Repertoire (sslDefault)	398	Video: Why Liberty? Rapid deployment and powerful administration	676
SSL Options (sslOptions)	398	Notices	678
TCP Options (tcpOptions)	398	Privacy Policy Considerations	681
Timed Operation (timedOperation)	400		
Transaction Manager (transaction)	401		
Trust Association Interceptor (trustAssociation)	454		
User Information (userInfo)	456		
Variable Declaration (variable).	457		
Virtual Host (virtualHost)	457		
Web Container Application Security (webAppSecurity)	462		
Web Application (webApplication)	466		
Web Container (webContainer)	473		
WAS WebSocket Outbound (wsocOutbound)	478		
z/OS Logging (zosLogging)	481		
Feature management	482		
Liberty features.	483		
Liberty Kernel	572		
Liberty Repository.	573		
Shared libraries.	576		
Product extension	577		
Security	581		
Quick overview of security	584		
Authentication	585		
Authorization	606		
Security on the Liberty application client container	608		
Java 2 Security	609		
Security public APIs	611		
Configuration differences between the traditional and Liberty: security	615		
The limits to protection through password encryption	616		
Java Persistence API (JPA)	617		
Java Persistence API (JPA) feature overview	618		
Java Persistence API 2.1 behavior changes	619		
Binary logging	621		
BinaryLog command options	625		
Configuring binary logging in Liberty	629		
Multimedia	630		
Video: Configure session cache management with Liberty and WebSphere eXtreme Scale	631		
Video: DevOps with WebSphere Liberty Server	637		
Video: Enabling IHS for Liberty Dynamic Routing	642		
Video: Getting started with the Server Configuration Tool for WebSphere Liberty	646		
Video: Google OpenID Connect for applications on WebSphere Liberty	647		
Video: Installing Liberty from a ZIP file	652		
Video: Java EE 7 in Liberty	654		
Video: OpenID Connect on Liberty	660		
Video: Setting up Admin Center	666		
		Chapter 2. Migrating applications to Liberty	683
		Migrating data access applications to Liberty	683
		Configuration differences between the traditional and Liberty: dataSource and jdbcDriver elements	683
		Configuration differences between the traditional and Liberty: connectionManager element	684
		Migrating a DB2 data source to Liberty.	685
		Migrating a Derby embedded data source to Liberty	687
		Chapter 3. Installing Liberty	689
		Installing and uninstalling Liberty using Installation Manager	690
		Installing and uninstalling Liberty on distributed operating systems	691
		Installing and uninstalling Liberty on IBM i operating systems	788
		Installing and uninstalling Liberty using downloaded files and archives	833
		Installing Liberty developer tools and (optionally) Liberty	835
		Installing Liberty by extracting a Java archive file	836
		Installing Liberty by extracting a ZIP archive file	842
		Upgrading Liberty installations	847
		Applying an interim fix to a Liberty archive installation	849
		Configuring the Liberty server to start as a job in the QWAS85 subsystem on IBM i	850
		Uninstalling Liberty application-serving environment from IBM i operating systems	851
		Installing Liberty Repository assets	852
		Installing assets using the installUtility command	853
		Installing assets using the featureManager command	864
		Installing assets using Installation Manager	872
		Installing assets by using developer tools	873
		Verifying the integrity of Liberty installation	875
		productInfo command	875
		Docker support in Liberty	877
		Accessing a remote Liberty server in a Docker container by using developer tools	877
		Liberty and Chef	878

Installing the OpenShift Cartridge for Liberty . . .	878	Displaying the server configuration in a merged view	944
Installing the IBM WebSphere Application Server Liberty Buildpack into a Cloud Foundry Environment	880	Viewing the schema documentation for the server configuration	945
Chapter 4. Setting up Liberty.	883	Generating a Liberty server dump using developer tools	945
Creating a Liberty server manually	883	Packaging a Liberty server by using developer tools	945
Creating a Liberty server by using developer tools	884	Adding a data source by using developer tools	946
Creating a remote Liberty server by using developer tools	887	Administering Liberty manually	946
Creating a workbench Liberty server in a Docker container by using developer tools	890	Customizing the Liberty environment	947
Directory locations and properties	894	Administering Liberty from the command line	949
Specifying Liberty bootstrap properties	897	Adding and removing Liberty features	968
Setting the default host name of a Liberty server	899	Using include elements, variables, and Ref tags in configuration files	969
Default port numbers	899	Controlling dynamic updates	975
Using virtual hosts	900	Configuring class loaders and libraries for Java EE applications	976
Isolating two applications from each other	900	Configuring libraries for OSGi applications	981
Isolating applications based on the requested host or port	901	Configuring JPA for Liberty	982
Restricting access based on originating endpoint	902	Configuring a web server plug-in for Liberty	985
Virtual hosts.	902	Configuring session persistence for Liberty	990
Preparing and running an application client	904	Configuring and deploying a basic JCA ResourceAdapter	991
Creating a Liberty application client manually	906	Configuring ManagedExecutorService instances	1012
Creating a Liberty application client with multiple client modules	907	Configuring thread context service instances	1015
Setting up the server-management environment for Liberty by using collectives.	907	Configuring managed scheduled executors	1017
Collective architecture	908	Configuring managed thread factories.	1019
Collective security	911	Connecting to Liberty by using JMX	1021
Configuring a Liberty collective	912	Establishing a JMX MBean Liberty server connection	1033
Overriding Liberty server host information	919	File transfer	1034
Registering host computers with a Liberty collective	921	Transferring files in a Liberty collective	1035
Setting the JAVA_HOME variable for Liberty collective members	923	Configuring binary logging in Liberty.	1041
Setting up RXA for Liberty collective operations	925	Administering the transaction service on Liberty	1043
Setting up a Liberty server to use Bluemix services	927	Administering data access resources on Liberty	1046
bluemixUtility command	927	Administering web applications on Liberty	1058
Configuring Liberty for Bluemix Cloudant services	931	Administering Contexts and Dependency Injection applications on Liberty.	1071
Configuring Liberty for Bluemix Watson services	932	Administering JavaMail on Liberty.	1074
Platform-as-a-service environment considerations for setting up Liberty.	934	Administering Liberty using Admin Center	1074
Chapter 5. Administering Liberty	937	Setting up Admin Center	1075
XML escape characters	937	Logging in to Admin Center	1078
Administering Liberty by using developer tools	938	Customizing the Toolbox	1079
Editing the Liberty configuration by using developer tools.	938	Editing server configuration files in Admin Center	1081
Specifying the Liberty configuration with dropins files by using developer tools	939	Exploring and managing resources with Admin Center	1083
Starting and stopping a server by using developer tools	940	Configuration updates	1090
Switching a Liberty Docker server between run and debug mode by using developer tools	940	Liberty and Chef	1091
Defining a utility project as a shared library	941	Including configuration information from external xml files in the server.xml file	1091
Exploring the runtime environment by using developer tools	943	Configuration element merging rules	1092
		Chapter 6. Extending Liberty	1095
		Developing a Liberty feature for Liberty	1095
		Developing a Liberty feature manually	1095
		Creating a Liberty feature by using developer tools	1106

Developing an OSGi bundle with simple activation	1111
Composing advanced features by using OSGi Declarative Services	1116
Advanced Configuration	1121
Providing an application endpoint	1130
Liberty SPI utilities	1131
Including protected features	1137
Locating OSGi applications	1138
Developing with the JNDI default namespace in a Liberty feature	1139
Developing a custom TAI as a Liberty feature	1141
Dynamic content management	1141
Packaging and installing Liberty features	1143
Provide product information for your feature extension	1143
Embedding Liberty in your applications	1144
Creating Liberty servers from custom configurations	1146

Chapter 7. Securing Liberty and its applications 1147

Getting started with security in Liberty	1147
Quick overview of security	1149
Setting up BasicRegistry and role mapping on Liberty	1150
Securing communications in Liberty	1151
Enabling SSL communication in Liberty	1152
Creating SSL certificates for your Liberty using the Utilities menu	1161
Creating SSL certificates from the command line	1161
Configuring your web application and server for client certificate authentication	1164
Setting up Liberty to run in SP800-131a	1165
Configuring an httpEndpoint to use an SSL configuration other than the default	1167
Authenticating users in Liberty	1168
Configuring a user registry for Liberty	1168
Configuring the authentication cache in Liberty	1183
Configuring a JAAS custom login module for Liberty	1184
Configuring a Java Authentication SPI for Containers (JASPIC) User Feature	1187
Configuring LTPA in Liberty	1188
OpenID	1189
OpenID Connect	1190
Configuring an OpenID Relying Party in Liberty	1193
Configuring SPNEGO authentication in Liberty	1195
Customizing SSO configuration using LTPA cookies in Liberty	1203
Configuring RunAs authentication in Liberty	1203
Configuring TAI in Liberty	1205
Configuring a custom form login page	1207
Configuring SAML Web Browser SSO in Liberty	1209
Using OpenID Connect.	1212
Authentication Filters	1252
Authorizing access to resources in Liberty	1254

Configuring authorization for applications in Liberty	1254
Configuring security authorization for Liberty servers on IBM i	1255
OAuth	1256
Configuring Common Secure Interoperability version 2 (CSIv2) in Liberty	1276
Configuring inbound CSIv2 in Liberty	1277
Configuring outbound CSIv2 in Liberty	1282
Configuring security for the Liberty application client container and its applications	1287
Enabling SSL communication for the Liberty application client container	1287
Configuring a JAAS programmatic login on the Liberty application client container	1289
Configuring a JAAS custom login module for the Liberty application client container	1291
Configuring Common Secure Interoperability version 2 (CSIv2) in the Liberty application client container	1292
Configuring Java Servlet 3.1 support for security	1294
Configuring secure JMX connection to Liberty	1296
Configuring web security related properties in Liberty	1297
Customizing SSO configuration using LTPA cookies in Liberty	1297
Configuring your web application and server for client certificate authentication	1298
Configuring the Liberty server to track logged out LTPA tokens	1299
Configuring authentication aliases for Liberty	1300
Configuring JAAS for database authentication	1300
Developing extensions to the Liberty security infrastructure	1301
Developing a custom TAI for Liberty	1301
Developing JAAS custom login modules for a system login configuration	1305
Developing a custom JASPIC authentication provider for Liberty	1309
Developing a Java Authorization Contract for Containers (JACC) Authorization Provider	1312
Developing a customPasswordEncryption Provider.	1314
Customizing an application login to perform an identity assertion by using JAAS	1316
Developing a custom user registry in Liberty	1317
Developing JAAS custom login modules for database authentication	1318
Developing a programmatic login for obtaining authentication data	1319
Developing a custom thread identity service	1320
Security considerations	1321
Securing Liberty by using HTTP Strict Transport Security (HSTS)	1321

Chapter 8. Developing applications in the Liberty environment 1323

Developing OSGi applications in Liberty	1323
Enable OSGi Applications with Java EE 7 technologies	1323

Enabling integration of OSGi application services	1324
Custom blueprint namespace handlers	1325
Developing WebSocket applications in Liberty	1326
WebSocket	1327

Chapter 9. Deploying applications in Liberty 1329

Adding and running an application on Liberty by using developer tools	1331
Publishing your application by using developer tools	1332
Restart requirements for a modified application on Liberty	1333
Customizing automatic feature detection	1334
Packaging a Liberty server from the command line	1334
Using JNDI binding for constants from the server configuration files	1336
Using JNDI binding for dynamic values from the server configuration files	1337
Deploying OSGi applications to Liberty	1338
Sharing common OSGi bundles for Liberty	1339
Deploying data access applications to Liberty	1339
Deploying an existing JDBC application to Liberty	1339
Enabling JDBC Tracing for Liberty	1342
Deploying a web application to Liberty	1344
Deploying SIP applications to Liberty	1346
Deploying a JPA application to Liberty	1347
Enhancement of JPA entities	1348
Deploying web services applications to Liberty	1348
Deploying JAX-RS 2.0 applications to Liberty	1348
Deploying Java batch applications for Liberty	1368
Java batch and managed batch overview	1368
Configuring Liberty for the batch REST API	1368
Java batch persistence configuration	1369
Securing the Liberty batch environment	1371
Java batch shutdown and recovery	1373
Batch REST API	1373
Enabling multiple server support by using the Liberty embedded messaging provider	1387
Enabling multiple server support by using the WebSphere MQ messaging provider	1389
Enabling multiple server partitions support by using the WebSphere MQ messaging provider	1393
Enabling multiple server partitions support by using the Liberty embedded messaging provider.	1399
Enabling batch job events publishing	1403
batchManager command-line client utility	1406
Viewing Java batch job logs	1408
Shared libraries	1410
Loose applications	1412
Discovering REST API documentation on a Liberty server	1416

Subscribe to Liberty REST API updates	1418
REST endpoints for pushing APIs into IBM API Connect	1419

Chapter 10. Monitoring the Liberty server runtime environment 1423

JVM monitoring	1423
Web application monitoring	1424
ThreadPool monitoring.	1425
SIP application monitoring	1426
Sessions monitoring.	1441
ConnectionPool monitoring	1442
Multiple components monitoring	1443
HTTP access logging	1444
HTTP access log settings	1444
HTTP access log format	1444

Chapter 11. Tuning Liberty 1445

Tuning Liberty for secure applications.	1447
Tuning federated LDAP repositories in Liberty	1449

Chapter 12. Troubleshooting tips 1451

Security bulletins for the Liberty profile	1457
Logstash collector	1457
Using the Logstash collector	1460
Logging and Trace	1461
Viewing trace and message log files by using developer tools	1465
Timed operations and JDBC calls	1466
Event Logging	1467
Slow and hung request detection	1469
Binary logging	1471
BinaryLog command options.	1475
Configuring binary logging in Liberty.	1479
Runtime environment known issues and restrictions	1480
Developer Tools known issues and restrictions	1488
Messages	1490
Troubleshooting OSGi applications by using developer tools	1494
Troubleshooting Session Initiation Protocol (SIP) on Liberty	1495
Troubleshooting the SIP container session repository on Liberty	1495
Tracing a Session Initiation Protocol (SIP) container on Liberty.	1498
Session Initiation Protocol (SIP) binary log and trace extensions on Liberty	1499

Chapter 13. Reference 1501

Programming Interfaces (APIs and SPIs)	1501
Messages	1501

Index 1503

Chapter 1. WebSphere Application Server Liberty Core: Overview

WebSphere® Application Server Liberty Core is a lightweight Liberty-based edition. With WebSphere Application Server Liberty Core, you can rapidly build and deliver web applications that do not require the full Java™ EE stack.

WebSphere Application Server Liberty Core is based on the developer-friendly WebSphere Application Server Liberty. By using WebSphere Application Server Liberty Core you can create applications corresponding to the Java EE6 Web Profile specification. You can quickly develop and deploy Web Profile-centric applications so that your business can respond quickly to enterprise and market needs. The capabilities that WebSphere Application Server Liberty Core provides are a subset of the capabilities that are provided in the WebSphere Application Server and WebSphere Application Server, Network Deployment editions.

WebSphere Application Server Liberty Core offers you the following benefits:

- An extremely lightweight edition, which contains the subset of Liberty that corresponds to the Java EE Web Profile specification.
- Excellent development and production runtime environments for web applications.
- A smaller footprint for faster download and startup, giving more development time and faster time to deployment.
- Ease of packaging applications for deployment, including configuration.
- The ability to run applications that are written for WebSphere Application Server Liberty Core on WebSphere Application Server traditional and Liberty.
- The ability to extend the Liberty capabilities by adding custom features that use a product extension System Programming Interface (SPI).

The WebSphere Application Server Liberty Core edition focuses on Web Profile capabilities such as servlet, JSP, JSF, and EJB-Lite. WebSphere Application Server Liberty Core has a different programming model from other WebSphere Application Server editions. For example, WebSphere Application Server Liberty Core does not include Java Message Service (JMS) or Web Services, making the Liberty Core edition lightweight and focused on the capabilities that are needed to deliver Web Profile applications.

You can use WebSphere Application Server Liberty Core to deploy to both development and production environments. The developer environment is provided by WebSphere Application Server Developer Tools for Eclipse (WDT). You can access a WebSphere Application Server Liberty Core management option through the WebSphere Application Server, Network Deployment job manager or Liberty-based collective controller.

Note: A WebSphere Application Server Liberty Core server can be a member of a collective, but a WebSphere Application Server, Network Deployment license is required for the collective controller.

Architecture

Liberty is a highly composable and dynamic runtime environment. OSGi services are used to manage component lifecycles, and the injection of dependencies and configuration. The server process comprises a single JVM, the Liberty kernel, and any number of optional features. The feature code and most of the kernel code runs as OSGi bundles within an OSGi framework. Features provide the programming models and services that are required by applications.

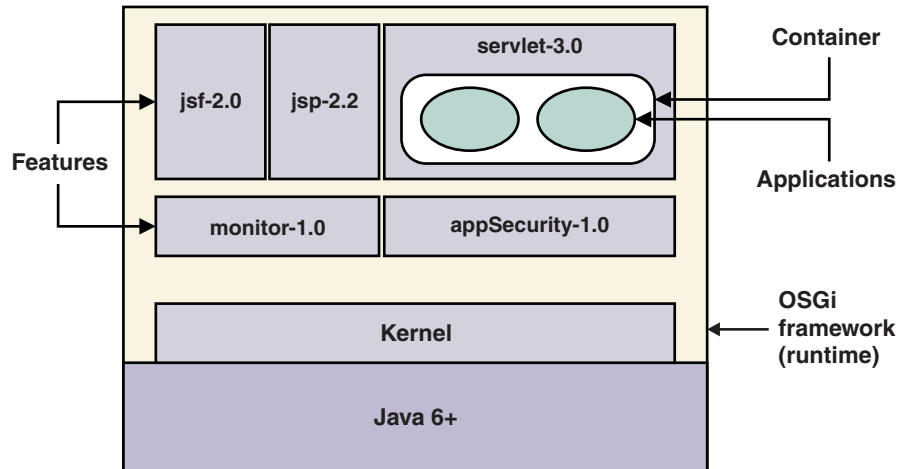


Figure 1. Liberty architecture

The kernel launcher bootstraps the system and starts the OSGi framework. The configuration is parsed, and then the configured features are loaded by the feature manager. The kernel extensively uses OSGi services to provide a highly dynamic runtime environment. The OSGi Configuration Admin service manages system configuration, and an OSGi Declarative Services component manages the lifecycle of system services. The file monitor service detects application and configuration file changes, and the logging service writes messages and debug information to the local file system.

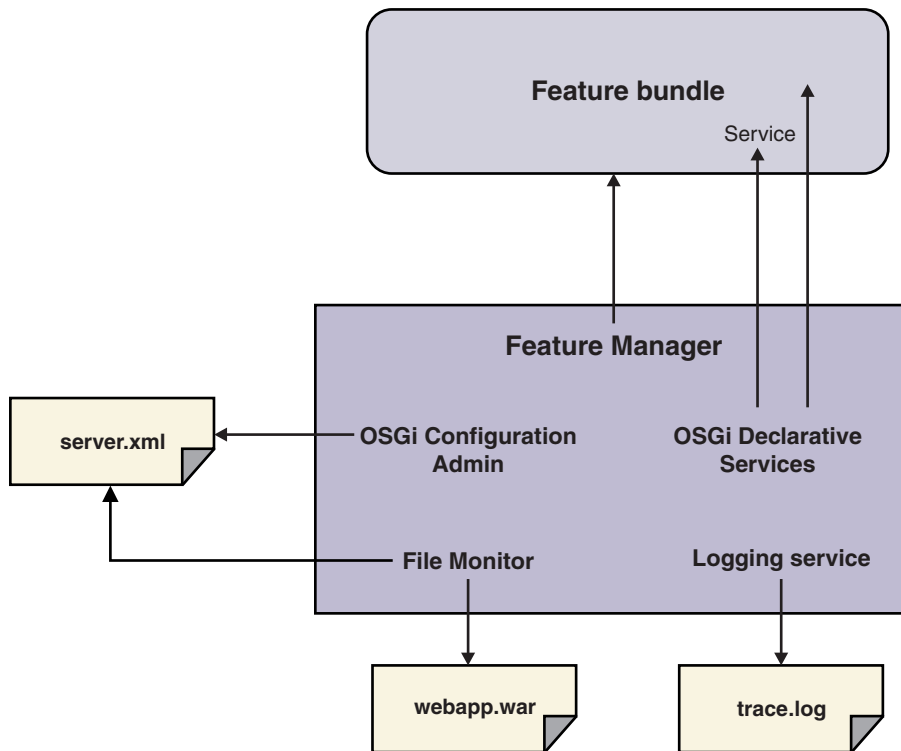


Figure 2. Liberty kernel

Features are specified in the system configuration files that are the `server.xml` file and any other included files. The server configuration files populate the OSGi Configuration Admin service, which injects the feature configuration into the feature manager service. The feature manager maps each feature name to a

list of bundles that provide the feature. The bundles are installed into the OSGi framework and started. The feature manager responds to configuration changes by dynamically adding and removing features while the server is running.

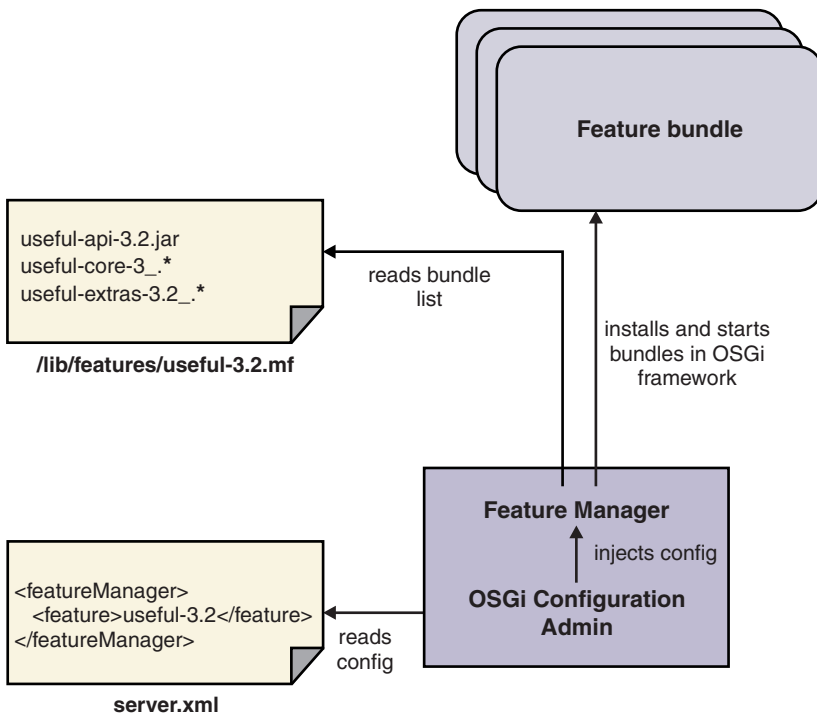


Figure 3. Feature management

Runtime services provide configuration default settings so that the configuration you need to specify is kept to a minimum. You specify the features you need, along with any additions or overrides to the system default settings, in a `server.xml` file. You might choose to structure your configuration into a number of separate files that are linked to the parent `server.xml` file by using an "include" syntax. At server startup, or when the user configuration files are changed, the kernel configuration management parses your configuration and applies it over the system default settings. The set of configuration properties that belongs to each service is injected into the service each time the configuration is updated.

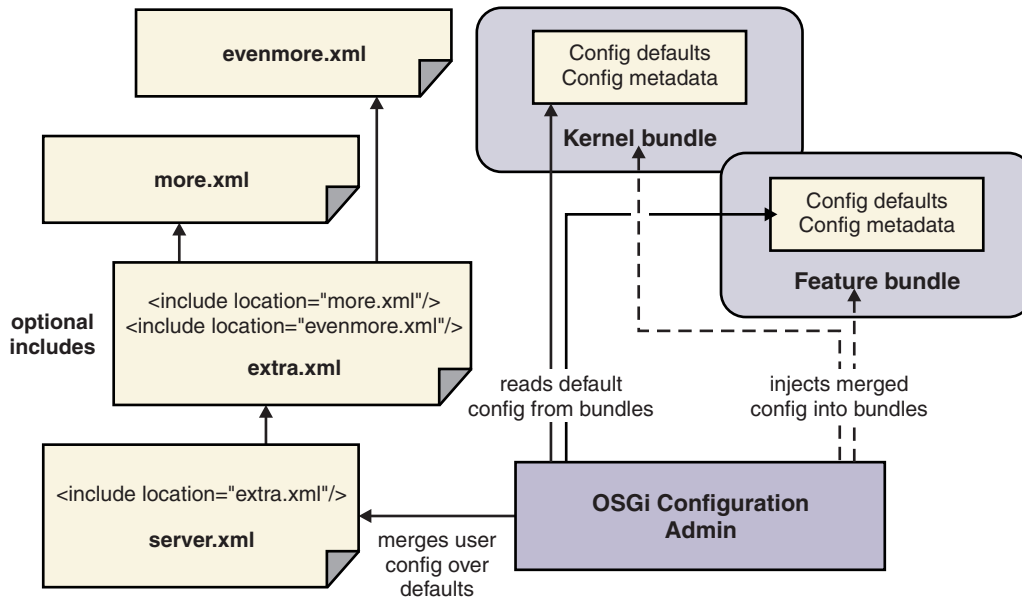


Figure 4. Configuration management

The OSGi Declarative Services component is used so that function can be decomposed into discrete services, which are activated only when needed. This behavior helps the runtime environment to be “late and lazy”, keeping the footprint small and the startup fast. Declared services are added to the OSGi service registry, and dependencies between services can be resolved without loading implementation classes. Service activation can be delayed until a service is used: when the service reference is resolved. Configuration for each service is injected as the service is activated, and is reinjected if the configuration is later modified.

Java EE 7 programming model support

Liberty complies with Java Platform, Enterprise Edition (Java EE) 7. The Java EE 7 table and links show the extent to which each of the major server profiles supports the full WebSphere Application Server programming model.

Java EE 7 technologies

Table 1. Java EE 7 support by profile.

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, the Liberty profile, and Liberty Core.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Java Platform, Enterprise Edition 7 (Java EE 7)	JSR 342	javaee-7.0 javaeeClient-7.0		8.5.5.6	
Java Platform, Enterprise Edition 7 Web Profile	JSR 342	webProfile-7.0		8.5.5.6	8.5.5.6
Web services technologies					

Table 1. Java EE 7 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, the Liberty profile, and Liberty Core.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Java API for RESTful Web Services (JAX-RS) 2.0	JSR 339	jaxrs-2.0		8.5.5.6	8.5.5.6
Implementing Enterprise Web Services 1.4	JSR 109			8.5.5.4	
Java API for XML-Based Web Services (JAX-WS) 2.2	JSR 224	jaxws-2.2			
Web Services Interoperability Organization (WS-I) Basic Profile	WS-I Basic Profile 1.2 WS-I Basic Profile 2.0	jaxws-2.2			
Java Architecture for XML Binding (JAXB) 2.2	JSR 222	jaxb-2.2			
Web Services Metadata for the Java Platform	JSR 181				
Java API for XML-based RPC (JAX-RPC) 1.1 (Optional)	JSR 101				
Java API for WSDL (JWSDL)	JSR 110				
SOAP with Attachments API for Java (SAAJ) 1.3(SOAP with Attachments API for Java (SAAJ) is also referred to as <i>Java APIs for XML Messaging</i> .)	JSR 67				
Java API for XML Registries (JAXR) 1.0 (Optional)	JSR 93				
Web application technologies					
Java API for JSON Processing (JSON-P) 1.0	JSR 353	jsonp-1.0		8.5.5.4	8.5.5.4
Java Servlet 3.1	JSR 340	servlet-3.1		8.5.5.4	8.5.5.4
JavaServer Faces (JSF) 2.2	JSR 344	jsf-2.2		8.5.5.6	8.5.5.6
JavaServer Pages 2.3	JSR 245	jsp-2.3		8.5.5.5	8.5.5.5

Table 1. Java EE 7 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, the Liberty profile, and Liberty Core.






























Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Expression Language (JSP/EL) 3.0	JSR 341	el-3.0		8.5.5.5 	8.5.5.5 
Standard Tag Library for JavaServer Pages (JSTL) 1.2	JSR 52				
Debugging Support for Other Languages 1.0	JSR 45				
WebSocket 1.1	JSR 356	websocket-1.1		8.5.5.5 	8.5.5.5 
WebSocket 1.0	JSR 356	websocket-1.0		8.5.5.4 	8.5.5.4 
Enterprise application technologies					
EE Concurrency Utilities 1.0	JSR 236	concurrent-1.0		8.5.5.4 	8.5.5.4 
Contexts and Dependency Injection for Java (Web Beans) 1.2	JSR 346	cdi-1.2		8.5.5.6 	8.5.5.6 
Contexts and Dependency Injection for Java (Web Beans) 1.1	JSR 346	cdi-1.2(Java EE 7 defines CDI 1.1. The CDI maintenance release CDI 1.2. The cdi-1.2 feature supports both CDI 1.1 and CDI 1.2.)		8.5.5.6 	8.5.5.6 
Dependency Injection for Java 1.0	JSR 330				
Bean Validation 1.1	JSR 349	beanValidation-1.1		8.5.5.6 	8.5.5.6 
Enterprise JavaBeans (EJB) 3.2 full	JSR 345	ejb-3.2(The ejb-3.2 feature includes the following EJB sub-features: ejbLite-3.2, ejbHome-3.2, ejbPersistentTimer-3.2, ejbRemote-3.2, and mdb-3.2.)		8.5.5.6 	
Enterprise JavaBeans (EJB) 3.2 Lite	JSR 345	ejbLite-3.2		8.5.5.6 	8.5.5.6 
Interceptors 1.2	JSR 318			8.5.5.6 	8.5.5.6 
Java EE Connector Architecture (JCA) 1.7	JSR 322	jca-1.7		8.5.5.6 	

Table 1. Java EE 7 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, the Liberty profile, and Liberty Core.
















Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Java Persistence 2.1	JSR 338	jpa-2.1		8.5.5.6 	8.5.5.6 
Common Annotations for the Java Platform 1.2 (Common Annotations 1.2 added the javax.annotation.Priority single annotation type, which Contexts and Dependency Injection 1.2 uses. For information about CDI 1.2, see "Contexts and Dependency Injection 1.2" on page 516.)	JSR 250			8.5.5.6 	8.5.5.6 
Java Message Service (JMS) API 2.0	JSR 343	jms-2.0		8.5.5.6 	
Java Transaction API (JTA) 1.2	JSR 907			8.5.5.6 	8.5.5.6 
JavaMail 1.5	JSR 919	javaMail-1.5		8.5.5.6 	8.5.5.6 
Batch Applications for Java Platform 1.0	JSR 352	batch-1.0		8.5.5.6 	
Management and security technologies					
Java Authentication Service Provider Interface for Containers (JASPIC) 1.1	JSR 196	jspic-1.1		8.5.5.6 	8.5.5.9 
Java Authorization Contract for Containers (JACC) 1.5	JSR 115	jacc-1.5		8.5.5.6 	8.5.5.9 
Java EE Application Deployment 1.2 (Optional)	JSR 88				

Table 1. Java EE 7 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, the Liberty profile, and Liberty Core.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
J2EE Management 1.1 (To invoke Management EJB APIs, the server configuration must have both the j2eeManagement-1.1 and ejbRemote-3.2 features in a feature manager. After both features are in the server configuration, you can invoke Management EJB API through JNDI name lookup. The Management EJB binding name (JNDI lookup name) is ejb/mejb/MEJB.)	JSR 77	j2eeManagement-1.1	△	8.5.5.6 △	
Java EE-related specifications in Java SE					
Java API for XML Processing (JAXP) 1.4	JSR 206		△	△	△
Java Database Connectivity (JDBC) 4.1	JSR 221	jdbc-4.1		8.5.5.5 △	8.5.5.5 △
Java Management Extensions (JMX) 2.0	JSR 255		△	△	△
JavaBeans Activation Framework (JAF) 1.1	JSR 925		△	△	△
Streaming API for XML (StAX) 1.0	JSR 173		△	△	△

Programming model extensions

For a list of WebSphere programming model extensions, see "WebSphere extensions" in the WebSphere Application Server: Overview and quick start topic.

Java EE 6 programming model support

The Java EE 6 table and links show the extent to which each of the major server profiles supports the full WebSphere Application Server programming model.

Java EE 6 technologies

Table 2. Java EE 6 support by profile.

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, by the Liberty profile, and by Liberty Core. The Liberty Core edition contains a subset of the Liberty features that are available in the other product editions.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Java Platform, Enterprise Edition 6 (Java EE 6)	JSR 316		△		
Java Platform, Enterprise Edition 6 Web Profile	JSR 316	webProfile-6.0	△	△	△
Web services technologies					
Java API for RESTful Web Services (JAX-RS) 1.1	JSR 311	jaxrs-1.1	△	△	△
Implementing Enterprise Web Services 1.4	JSR 109		△	△	
Java API for XML-Based Web Services (JAX-WS) 2.2	JSR 224	jaxws-2.2	△	△	
Web Services Interoperability Organization (WS-I) Basic Profile	WS-I Basic Profile 1.2 WS-I Basic Profile 2.0	jaxws-2.2	△	△	
Java Architecture for XML Binding (JAXB) 2.2	JSR 222	jaxb-2.2	△	△	
Web Services Metadata for the Java Platform	JSR 181		△	△	
Java API for XML-based RPC (JAX-RPC) 1.1	JSR 101		△		
Java API for WSDL (JWSDL)	JSR 110		△	△	
SOAP with Attachments API for Java (SAAJ) 1.3(SOAP with Attachments API for Java (SAAJ) is also referred to as <i>Java APIs for XML Messaging</i> .)	JSR 67		△	△	
Java API for XML Registries (JAXR) 1.0	JSR 93		△		

Table 2. Java EE 6 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, by the Liberty profile, and by Liberty Core. The Liberty Core edition contains a subset of the Liberty features that are available in the other product editions.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
Web application technologies					
Java Servlet 3.0	JSR 315	servlet-3.0	△	△	△
JavaServer Faces (JSF) 2.0	JSR 314	jsf-2.0	△	△	△
JavaServer Pages 2.2/Expression Language (JSP/EL) 2.2	JSR 245	jsp-2.2	△	△	△
Standard Tag Library for JavaServer Pages (JSTL) 1.2	JSR 52		△	△	△
Debugging Support for Other Languages 1.0	JSR 45		△	△	△
Enterprise application technologies					
Contexts and Dependency Injection for Java (Web Beans 1.0)	JSR 299	cdi-1.0	△	△	△
Dependency Injection for Java 1.0	JSR 330		△	△	△
Bean Validation 1.0	JSR 303	beanValidation-1.0	△	△	△
Enterprise JavaBeans (EJB) 3.1 (includes Interceptors 1.1)	JSR 318	ejbLite-3.1	△	△(The Liberty supports only the EJB Lite subset and Message Driven Beans. See the "Enterprise JavaBeans (EJB) Lite subset" section of "Liberty features" on page 483.)	△(Liberty supports only the EJB Lite subset and Message Driven Beans. See the "Enterprise JavaBeans (EJB) Lite subset" section of "Liberty features" on page 483.)
Java EE Connector Architecture 1.6	JSR 322	jca-1.6	△	8.5.5.2 △	
Java Persistence 2.0	JSR 317		△	△	△
Common Annotations for the Java Platform 1.1	JSR 250		△	△	△
Java Message Service (JMS) API 1.1	JSR 914	jms-1.1	△	△	
Java Transaction API (JTA) 1.1	JSR 907		△	△	△

Table 2. Java EE 6 support by profile (continued).

A list of Java EE technologies, subdivided into sections for web services, web applications, enterprise applications, management and security, and Java EE-related specifications in Java SE. For each technology there is a specification reference, any related Liberty feature, and an indication of whether the technology is supported by the full profile, by the Liberty profile, and by Liberty Core. The Liberty Core edition contains a subset of the Liberty features that are available in the other product editions.

Technology	Specification reference	Liberty feature	Full profile	Liberty profile	Liberty Core
JavaMail 1.4	JSR 919		△		
Management and security technologies					
Java Authentication Service Provider Interface for Containers (JASPIC)	JSR 196		△		
Java Authorization Contract for Containers (JACC) 1.3	JSR 115		△		
Java EE Application Deployment 1.2	JSR 88		△		
J2EE Management 1.1	JSR 77		△		
Java EE-related specifications in Java SE					
Java API for XML Processing (JAXP) 1.4	JSR 206		△	△	△
Java Database Connectivity (JDBC) 4.0	JSR 221	jdbc-4.0	△	△	△
Java Management Extensions (JMX) 2.0	JSR 255		△	△	△
JavaBeans Activation Framework (JAF) 1.1	JSR 925		△	△	△
Streaming API for XML (StAX) 1.0	JSR 173		△	△	△

Programming model extensions

For a list of WebSphere programming model extensions, see "WebSphere extensions" in the WebSphere Application Server: Overview and quick start topic.

Supported Java EE 6 and 7 feature combinations

8.5.5.6

Some combinations of Java EE 7 and Java EE 6 Liberty features in a server configuration are compatible. However, many combinations are not compatible and cause an error when the server starts.

The error message resembles:

CWWKF0033E: The singleton features com.ibm.websphere.appserver.javaeeCompatible-6.0 and com.ibm.websphere.appserver.javaeeCompatible-7.0 are incompatible.

The following table marks compatible feature combinations with a checkmark (Δ). Ensure that your server configuration does not contain incompatible features.

Table 3. Supported combinations of Java EE 7 and Java EE 6 Liberty features. Java EE 7 features are listed vertically. Java EE 6 features are listed horizontally. A checkmark (Δ) indicates that the combination of Java EE 7 and 6 features is supported and a server configuration can contain both features. An empty cell (no Δ) indicates that the combination of Java EE 7 and 6 features is not supported.

Java EE 7 features	Java EE 6 features														
	beanValidation-1.0	cdi-1.0	ejbLite-3.1	jaxb-2.2	jaxrs-1.1	jaxws-2.2	jca-1.6	jdbc-4.0	jms-1.1	jpa-2.0	jsf-2.0	jsp-2.2	managed-Beans-1.0	mdb-3.1	servlet-3.0
batch-1.0				Δ		Δ		Δ		Δ			Δ		
beanValidation-1.1				Δ		Δ		Δ		Δ			Δ		
cdi-1.2				Δ		Δ		Δ		Δ			Δ		
concurrent-1.0	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
el-3.0	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
ejb-3.2				Δ		Δ				Δ			Δ		
ejbLite-3.2				Δ		Δ		Δ		Δ			Δ		
javaMail-1.5	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
jacc-1.5	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
jaspic1.1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
jaxrs-2.0				Δ		Δ		Δ		Δ			Δ		
jca-1.7				Δ		Δ		Δ		Δ			Δ		
jdbc-4.1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
jms-2.0				Δ		Δ		Δ		Δ			Δ		
jpa-2.1				Δ		Δ							Δ		
jsf-2.2				Δ		Δ		Δ		Δ			Δ		
jsonp-1.0	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
jsp-2.3	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
mdb-3.2				Δ		Δ		Δ		Δ			Δ		
servlet-3.1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
websocket-1.0	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
websocket-1.1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ

The table shows that the `servlet-3.0` feature of Java EE 6 is incompatible with the `websocket-1.1` feature of Java EE 7. Thus, a server configuration with the following features causes an error:

```
<featureManager>
  <feature>servlet-3.0</feature>
  <feature>websocket-1.1</feature>
</featureManager>
```

To resolve the error, use `servlet-3.1` instead of `servlet-3.0` in the server configuration. The `servlet-3.1` feature is compatible with the `websocket-1.1` feature.

For more information on compatibility, or toleration, of features, see [Tolerating features](#).

Java EE 7 behavior changes

If you previously incorporated Java Platform, Enterprise Edition (Java EE) 6 features in your Liberty environment, you might encounter behavior changes when you move to a Java EE 7 feature.

You can choose between the Java EE 6 and Java EE 7 feature implementations for each server instance, with consideration for behavior changes. If the required behavior is contained in the Java EE 7 feature only, you must use the Java EE 7 feature. If an existing application would be adversely impacted by behavior changes in the Java EE 7 feature, using the Java EE 6 feature preserves the existing behavior for that application. You must ensure that the Java EE implementation that you choose is compatible with other Java EE features in your server; for more information, see “Supported Java EE 6 and 7 feature combinations” on page 11.

Table 4. Features that have Java EE 6 and 7 implementations

Technology	Java EE 6 feature	Java EE 7 feature	Behavior changes
Bean Validation	beanValidation-1.0	beanValidation-1.1	No behavior changes.
Contexts and Dependency Injection for Java (CDI)	cdi-1.0	cdi-1.2	See “Contexts and Dependency Injection 1.2 behavior changes” on page 1072.
Enterprise JavaBeans (EJB)	ejbLite-3.1	ejbLite-3.2	No behavior changes.
Expression Language (EL)	Included as part of jsp-2.2	el-3.0	See “Expression Language 3.0 feature functions” on page 1068.
Java API for RESTful Web Services (JAX-RS)	jaxrs-1.1	jaxrs-2.0	See “JAX-RS 2.0 behavior changes” on page 1357.
Java Database Connectivity (JDBC)	jdbc-4.0	jdbc-4.1	No behavior changes.
Java Persistence API (JPA)	jpa-2.0	jpa-2.1	See “Java Persistence API 2.1 behavior changes” on page 619.
Java Servlet	servlet-3.0	servlet-3.1	See “Servlet 3.1 behavior changes” on page 1060.
Java Transaction API (JTA)	transaction-1.1 (protected feature)	transaction-1.2 (protected feature)	No behavior changes.
JavaServer Faces (JSF)	jsf-2.0	jsf-2.2	See “Configuring Liberty for JavaServer Faces 2.2” on page 1068.
JavaServer Pages (JSP)	jsp-2.2	jsp-2.3	No behavior changes.

Enterprise OSGi programming model support

The Enterprise OSGi table and links show the extent to which each of the major server profiles supports the full WebSphere Application Server programming model.

Enterprise OSGi technologies

Table 5. Enterprise OSGi support by profile.

A list of enterprise OSGi technologies, subdivided into sections for blueprint, web, and other enterprise technologies. For each technology there is a specification reference, and an indication of whether the technology is supported by the full profile, by the Liberty profile, and by Liberty Core.

Technology	Specification reference	Full profile	Liberty profile	Liberty Core
Blueprint-related technologies				
Blueprint Container	R4.2 Enterprise Chapter 121	△	△	△
Blueprint Transactions		△	△	△
Blueprint Managed JPA		△	△	△
Blueprint Security		△		
Blueprint Resource References		△		
Custom Blueprint Namespaces			8.5.5.4 △	8.5.5.4 △
Web-related technologies				
Web Application Bundles	R4.2 Enterprise Chapter 128	△	△	△

Table 5. Enterprise OSGi support by profile (continued).

A list of enterprise OSGi technologies, subdivided into sections for blueprint, web, and other enterprise technologies. For each technology there is a specification reference, and an indication of whether the technology is supported by the full profile, by the Liberty profile, and by Liberty Core.

Technology	Specification reference	Full profile	Liberty profile	Liberty Core
8.5.5.7 WebSocket Applications			8.5.5.7 △	
JNDI	R4.2 Enterprise Chapter 126	△	△	△
JSP		△	△	△
JSTL		△	△	△
JSF		△	△	△
JAX-RS		△	△	△
Other enterprise technologies				
EJB Bundles		△(EJB levels earlier than 3.0 are not supported.)		
Remote Services	R4.2 Compendium Chapter 13	△		
SCA Configuration Type Specification	R4.2 Enterprise Chapter 129	△		
Remote Bundle Repositories		△	△	△
SIP		△(SIP annotations are not supported.)		
Local OSGi application integration			8.5.5.5 △	8.5.5.5 △

Note: **8.5.5.7** WebSockets is currently only supported on Liberty.

Liberty externals support

External functions and resources of Liberty can be used directly, and can be relied on to be in the next release. Internal or incidental aspects of Liberty might change when you apply service, or upgrade to a future release.

What can I use directly in Liberty and rely on being in the next release?

The following resources can be used directly and will continue to be available in the next release:

- The application programming interfaces (APIs) and system programming interfaces (SPIs) defined by the content of the JAR files in the `${wlp.install.dir}/dev` directories.
 - The application class loader has visibility to the API that is provided by the features in your server configuration. Product extension features have visibility to all API and SPI that is provided by the features in your server configuration.
 - Compile your code against the JAR files in the `${wlp.install.dir}/dev` directories. The JAR files in the `${wlp.install.dir}/dev` directories are provided only for compilation of applications and features, they are not supported for runtime use. Do not use these JAR files in applications, libraries, or tests.
- The server configuration, including features with *public* or *protected* visibility. Public features and configuration elements can be specified in the `server.xml` file and included files; protected features can be included in your own features.
- Commands, scripts and archives in the `${wlp.install.dir}/bin` directory and subdirectories.
- Client utilities in the `${wlp.install.dir}/clients` directory and subdirectories.

What should I avoid dependencies on?

Do not build dependencies on incidental aspects of the product, or you might be impacted when you apply service or upgrade to future releases. Examples of product internals that you should avoid relying on include, but are not restricted to, the following scenarios:

- The names of product binary jars, for example those in the `${wlp.install.dir}/dev` directory. Compile your code against these JAR files by using the tools or the **javac -extdirs** option.

8.5.5.4 If you are using Apache Ant to compile your code, use wildcards to avoid dependencies on the specific JAR version; for example:

```
<fileset dir="${wlp.install.dir}/dev/api/spec" includes="com.ibm.ws.javaee.servlet.3.0_*.jar"/>
```

Alternatively, you can use the **featureManager classpath** command to generate a classpath for a specific set of features. See “Overriding classes from the Java SDK” on page 1482.

- Direct use of the product binaries in the `${wlp.install.dir}/lib` directory. The only JAR files that can be directly invoked are in the `${wlp.install.dir}/bin/tools` directory.
- Messages that are output by the server at run time. The text and inserts of messages are subject to change in service and version upgrades. As far as practically possible, the product will be consistent in the message IDs that are output at particular points of operation, but this cannot be guaranteed because underlying implementations might change.
- The layout of the product installation, other than the `${wlp.install.dir}/bin` and `${wlp.install.dir}/dev` directories.
- Examples and template files in the `${wlp.install.dir}/templates` directory. These files might be modified when you apply services to your installation.
- Private or third party Java packages that are not explicitly exposed as APIs. These are not visible to the application class loader at run time.
- Do not use the `console.log` file for the automated processing of server output. Instead, use the `messages.log` file for accessing and processing the messages, which provides more detailed information, in a format that is easier to process.

What might be modified by applying service or an upgrade?

The contents of the following directories and their subdirectories might be modified when service or upgrade is applied. Do not make your own modifications to files in these locations, or they might be overwritten by product maintenance or upgrade:

- `${wlp.install.dir}/bin`
- `${wlp.install.dir}/clients`
- `${wlp.install.dir}/dev`
- `${wlp.install.dir}/java`
- `${wlp.install.dir}/lib`
- `${wlp.install.dir}/templates`

No modifications are made to the contents of the following directories. These are your files, and applying service or upgrade will not modify them:

- `${wlp.install.dir}/etc` (where you might have added a `server.env` or `jvm.options` file).
- `${wlp.install.dir}/usr` (the default location for user configuration and applications).
- Any non-default directory that you designate through the `WLP_USER_DIR` environment variable.

IBM i There is an exception to the policy that no modifications are made to the contents of `${wlp.install.dir}/etc`. The file `${wlp.install.dir}/etc/default.env` is created when you install Liberty on IBM® iSeries Platforms using the Installation Manager. This file is also created or replaced by

the iAdmin POSTINSTALL command during archive and Job Manager installations. The iAdmin command is in the `${wlp.install.dir}/lib/native/os400/bin` directory. See “iAdmin command” on page 953.

Third-party APIs might change over time without consideration to backward compatibility. These are Java packages that are considered part of the implementation of features developed in open source communities and delivered as part of Liberty. Third-party APIs are not visible to applications by default; Java EE applications with a classloader configuration that explicitly allows third-party access will have visibility to those packages on the application class loader, and OSGi applications must explicitly import the packages. Consider the impact of incompatible changes before deciding to use third-party APIs.

Server configuration

Liberty is configured by exception. The runtime environment operates from a set of built-in configuration default settings, and you only need to specify configuration that overrides those default settings. You do this by editing either the `server.xml` file or another XML file that is included in `server.xml` at run time.

The configuration has the following characteristics:

- Described in XML files.
- Human-readable, and editable in a text editor.
- Small, easy to back up, and easy to copy to another system.
- Shareable across an application development team.
- Composable, so that features can easily add their own configuration to the system.
- Extensibly-typed, so you don't have to modify the current configuration to work with later versions of the runtime environment.
- Dynamically responsive to updates.
- Forgiving, so that missing values are assumed and unrecognized properties are ignored.

Features are the units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server. They are the primary mechanism that makes the server composable. The list of features that you specify in the server configuration provides a functional server. See “Liberty features” on page 483.

When you first install and start the server, a feature manager and a default server configuration are available:

- By default, a server contains the `jsp-2.2` feature, to support servlet and JSP applications. You can use the feature manager to add the features that you need.
- Server configuration is by exception. When you specify the features that you need, the default configuration of those features provides a rich environment that is designed to cover most common requirements, therefore you only need to specify changes from the default configuration.

For a full list of the elements that you can configure to complement or modify the configuration provided by Liberty features, see `**** MISSING FILE ****`.

You can also use a `bootstrap.properties` file to specify properties that are needed before the main configuration is processed, and to define variables that are used in the main configuration.

For a complete list of configuration files, see `Directory locations and properties`.

Service author perspective: Runtime management of configuration

The Liberty configuration service parses the primary `server.xml` file and any files it includes.

8.5.5.5 The Liberty configuration service also parses the configuration files in the configDropins directory.

The Liberty configuration service parses the files, merges the contents over the default configuration values provided by the installed bundles, then feeds the resulting property sets into the OSGi Configuration Admin Service (CA). CA injects each set of properties into the service that owns the set, if it is registered with CA.

The ordering of these steps is flexible. Services can register with CA before or after the initial property sets are established. Properties can be updated in CA after the initial injection, at which time the updated properties are injected into the owning service. It is therefore important that the services can receive, and respond appropriately to, updates to their configuration at any time that the service is active. Specifically, if a service delays its activation until its configuration is available, it must still be able to activate.

To enable a service to receive configuration data, there are a number of steps involved. See “Enabling a service to receive configuration data” on page 1118.

Microsoft Active Directory LDAP Filters (activatedLdapFilterProperties)

Specifies the list of default Microsoft Active Directory LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectcategory=group))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	memberOf:member	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(sAMAccountName=%v)(objectcategory=user))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	user:sAMAccountName	An LDAP filter that maps the name of a user to an LDAP entry.

Administrator Role (administrator-role)

A collection of users and/or groups assigned the server administrator role.

- group
- user

group

Group assigned a role.

false

string

user

User assigned a role.

false

string

API Discovery (apiDiscovery)

Configuration for the API Discovery feature to document your REST APIs.

- webModuleDoc

Attribute name	Data type	Default value	Description
apiName	string		The name of the aggregated API.
id	string		A unique configuration ID.
maxSubscriptions	int Minimum: 0 Maximum: 100	20	Specifies the maximum number of concurrent REST API clients that are listening for updates.

webModuleDoc

Configuration for each web module that provides API documentation to be exposed.

false

Attribute name	Data type	Default value	Description
contextRoot	string		The context root of the web module for which you are providing documentation.
docURL	string		The URL of the documentation of this web module. The URL can be relative to the context root by starting with a forward slash (/) or absolute by starting with http or https.
enabled	boolean	true	A boolean that controls the processing of documentation for this web module.
id	string		A unique configuration ID.

Application (application)

Defines the properties of an application.

- application-bnd
 - security-role
 - group
 - run-as
 - special-subject
 - user
- classloader
 - commonLibrary
 - file
 - fileset
 - folder
 - privateLibrary

- file
- fileset
- folder
- resourceAdapter
 - contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - customize

Attribute name	Data type	Default value	Description
autoStart	boolean	true	Indicates whether or not the server automatically starts the application.
context-root	string		Context root of an application.
id	string		A unique configuration ID.
location	A file, directory or url.		Location of an application expressed as an absolute path or a path relative to the server-level apps directory.
name	string		Name of an application.
suppressUncoveredHttpMethodWarning	boolean	false	Option to suppress uncovered HTTP method warning message during application deployment.
type	string		Type of application archive.

application-bnd

Binds general deployment information included in the application to specific resources.

false

Attribute name	Data type	Default value	Description
version	string		Version of the application bindings specification.

application-bnd > security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a security role.

application-bnd > security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		Group access ID
id	string		A unique configuration ID.
name	string		Name of a group possessing a security role.

application-bnd > security-role > run-as

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of a user required to access a bean from another bean. The value can be stored in clear text or encoded form. To encode the password, use the securityUtility tool with the encode option.
userid	string		ID of a user required to access a bean from another bean.

application-bnd > security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		<p>One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE.</p> <p>EVERYONE Everyone</p> <p>ALL_AUTHENTICATED_USERS All authenticated users</p>

application-bnd > security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user possessing a security role.

classloader

Defines the settings for an application classloader.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
classProviderRef	List of references to top level resourceAdapter elements (comma-separated string).		List of class provider references. When searching for classes or resources, this class loader will delegate to the specified class providers after searching its own class path.
commonLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are shared with other classloaders.
delegation	<ul style="list-style-type: none">parentFirstparentLast	parentFirst	Controls whether parent classloader is used before or after this classloader. If parent first is selected then delegate to immediate parent before searching the classpath. If parent last is selected then search the classpath before delegating to the immediate parent. parentFirst Parent first parentLast Parent last

Attribute name	Data type	Default value	Description
privateLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

classloader > commonLibrary

List of library references. Library class instances are shared with other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > commonLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > commonLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.

Attribute name	Data type	Default value	Description
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > commonLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

classloader > privateLibrary

List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > privateLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > privateLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > privateLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

resourceAdapter

Specifies configuration for a resource adapter that is embedded in an application.

false

Attribute name	Data type	Default value	Description
alias	string	#{id}	Overrides the default identifier for the resource adapter. The identifier is used in the name of the resource adapter's configuration properties element, which in turn is used in determining the name of configuration properties elements for any resources provided by the resource adapter. The resource adapter's configuration properties element name has the format, properties.<APP_NAME>.<ALIAS>, where <APP_NAME> is the name of the application and <ALIAS> is the configured alias. If unspecified, the alias defaults to the module name of the resource adapter.
autoStart	boolean		Configures whether a resource adapter starts automatically upon deployment of the resource adapter or lazily upon injection or lookup of a resource.
contextServiceRef	A reference to top level contextService element (string).		Configures how context is captured and propagated to threads.
id	string		Identifies the name of the embedded resource adapter module to which this configuration applies.

resourceAdapter > contextService

Configures how context is captured and propagated to threads.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

resourceAdapter > contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

resourceAdapter > contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

resourceAdapter > contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

resourceAdapter > contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

resourceAdapter > contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

resourceAdapter > contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

resourceAdapter > contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

resourceAdapter > contextService > classloaderContext

Classloader context propagation configuration.

false

resourceAdapter > contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

resourceAdapter > contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

resourceAdapter > contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

resourceAdapter > contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

resourceAdapter > customize

Customizes the configuration properties element for the activation specification, administered object, or connection factory with the specified interface and/or implementation class.

false

Attribute name	Data type	Default value	Description
implementation	string		Fully qualified implementation class name for which the configuration properties element should be customized.
interface	string		Fully qualified interface class name for which the configuration properties element should be customized.

Attribute name	Data type	Default value	Description
suffix	string		Overrides the default suffix for the configuration properties element. For example, "CustomConnectionFactory" in properties.rarModule1.CustomConnectionFa The suffix is useful to disambiguate when multiple types of connection factories, administered objects, or endpoint activations are provided by a resource adapter. If a configuration properties element customization omits the suffix or leaves it blank, no suffix is used.

Application Manager (applicationManager)

Properties that control the behavior of the application manager

Attribute name	Data type	Default value	Description
autoExpand	boolean	false	Enables automatic extraction of WAR files and EAR files
startTimeout	A period of time with second precision	30s	Specifies how long the server waits for an application to start before it considers it slow. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
stopTimeout	A period of time with second precision	30s	Specifies how long the server waits for an application to stop before it considers it slow. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Application Monitoring (applicationMonitor)

Defines how the server responds to application additions, updates, and deletions.

Attribute name	Data type	Default value	Description
dropins	Path to a directory	dropins	Location of the application drop-in directory expressed as an absolute path or a path relative to the server directory.
dropinsEnabled	boolean	true	Monitor the drop-in directory for application additions, updates, and deletions.
pollingRate	A period of time with millisecond precision	500ms	Rate at which the server checks for application additions, updates, and deletions. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
updateTrigger	<ul style="list-style-type: none"> • mbean • polled • disabled 	polled	<p>Application update method or trigger.</p> <p>mbean Server will only update applications when prompted by an MBean called by an external program such as an integrated development environment or a management application.</p> <p>polled Server will scan for application changes at the polling interval and update any applications that have detectable changes.</p> <p>disabled Disables all update monitoring. Application changes will not be applied while the server is running.</p>

Authentication Cache (authCache)

Controls the operation of the authentication cache.

Attribute name	Data type	Default value	Description
allowBasicAuthLookup	boolean	true	Allow lookup by user ID and hashed password.
initialSize	int Minimum: 1	50	Initial number of entries supported by the authentication cache.
maxSize	int Minimum: 1	25000	Maximum number of entries supported by the authentication cache.
timeout	A period of time with millisecond precision	600s	Amount of time after which an entry in the cache will be removed. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Authentication Data (authData)

Authentication alias for a connection to an Enterprise Information System (EIS) or database.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

Authentication Filter (authFilter)

Specifies a selection rule that represents conditions that are matched against the HTTP request headers to determine whether or not the HTTP request is selected for the authentication.

- host
- remoteAddress
- requestUrl
- userAgent

- webApp

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

host

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

remoteAddress

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ip	string		Specifies the IP address.
matchType	<ul style="list-style-type: none"> • lessThan • equals • greaterThan • contains • notContain 	contains	Specifies the match type. lessThan Less than equals Equals greaterThan Greater than contains Contains notContain Not contain

requestUrl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
urlPattern	string		Specifies the URL pattern.

userAgent

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
agent	string		Specifies the user agent
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain

webApp

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

Authentication (authentication)

Controls the built-in authentication service configuration.

Attribute name	Data type	Default value	Description
allowHashtableLoginWithIdOnly	boolean	false	Allow an application to login with just an identity in the hashtable properties. Use this option only when you have applications that require this and have other means to validate the identity.
cacheEnabled	boolean	true	Enables the authentication cache.

Feature Authorization Role Mapping (authorization-roles)

A collection of role names and mappings of the roles to users, groups, or special subjects

- security-role
 - group
 - special-subject
 - user

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Role name.

security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A group access ID in the general form group:realmName/groupUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a group that has the security role.

security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none">• EVERYONE• ALL_AUTHENTICATED_USERS		One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE. EVERYONE All users for every request, even if the request was not authenticated. ALL_AUTHENTICATED_USERS All authenticated users.

security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user who has the security role.

Basic User Registry (basicRegistry)

A simple XML-based user registry.

- group
 - member
- user

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ignoreCaseForAuthentication	boolean	false	Allow case-insensitive user name authentication.
realm	string	BasicRegistry	The realm name represents the user registry.

group

A group in a Basic User Registry.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a group in a Basic User Registry.

group > member

A member of a Basic User Registry group.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a user in a Basic User Registry group.

user

A user in a Basic User Registry.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a user in a Basic User Registry.
password	One way hashable, or reversably encoded password (string)		Password of a user in a Basic User Registry. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.

BELL (bell)

This feature enables the configuration of Basic Extensions using Liberty Libraries (BELL). A BELL enables the server runtime to be extended using shared libraries.

- library
 - file
 - fileset
 - folder
- service

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
libraryRef	A reference to top level library element (string).		The library to use for the BELL.

library

The library to use for the BELL.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

service

The name of the service that the system will look up in the /META-INF/services folder. If not specified, the system discovers all services located in the META-INF/services folder.

false

string

OSGi Applications Bundle Repository (bundleRepository)

An internal bundle repository, in which you can store the bundles for your OSGi applications.

- fileset

Attribute name	Data type	Default value	Description
filesetRef	List of references to top level fileset elements (comma-separated string).		Space separated list of fileset references
id	string		A unique configuration ID.
location	A file, directory or url.		Location of the remote repository expressed as an absolute URL or one relative to the server home directory.

fileset

Space separated list of fileset references

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Contexts And Dependency Injection (CDI) V1.2 (cdi12)

Defines the behavior of the Contexts and Dependency Injection (CDI) v1.2.

Attribute name	Data type	Default value	Description
enableImplicitBeanArchives	boolean	true	The implicit bean archives are scanned for any bean discoveries.

CDI Container (cdiContainer)

Defines behavior for the Contexts and Dependency Injection (CDI) container.

Channel Framework (channelfw)

Defines channel and chain management settings.

Attribute name	Data type	Default value	Description
chainQuiesceTimeout	A period of time with millisecond precision	30s	Default amount of time to wait while quiescing chains. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
chainStartRetryAttempts	int Minimum: 0	60	Number of retry attempts to make per chain.
chainStartRetryInterval	A period of time with millisecond precision	5s	Time interval between start retries. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
warningWaitTime	A period of time with millisecond precision	10s	Amount of time to wait before notifying of a missing factory configuration. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Classloading (classloading)

Global classloading

Attribute name	Data type	Default value	Description
useJarUrls	boolean	false	Whether to use jar: or wsjar: URLs for referencing files in archives

Certificate Authority Signed Certificate (collectiveCertificate)

Certificate authority signed certificate for collective.

Attribute name	Data type	Default value	Description
rdn	string	DC=com.ibm.ws.collective	Configure expected RDN attribute for incoming collective certificate.

Collective Member (collectiveMember)

The collective member configuration requires at least one collective controller address (identified by controllerHost and controllerPort). The collective controller can have multiple available addresses. Add one or more failoverController elements to identify the additional controllers. When multiple controllers are available, the member connects to one of the controllers. If the connection to the controller ends unexpectedly, the member connects to another controller in the remaining set of controllers.

- failoverController

Attribute name	Data type	Default value	Description
controllerHost	string		The host name for the collective controller instance.
controllerPort	int		The port for the JMX/REST connector, typically the HTTPS port.
controllerReadTimeout	A period of time with millisecond precision	300s	The read timeout for member connection to the collective controller. A longer read timeout may be necessary in large or geographically dispersed topologies. Minimum value is 2 minutes. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
heartBeatInterval	A period of time with millisecond precision	60s	Periodic time interval at which the collective member will contact the collective controller to indicate liveness. Minimum value is 1 second. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

failoverController

An additional collective controller address which is available for the collective member to use.

false

Attribute name	Data type	Default value	Description
host	string		The host name for the collective controller instance.
id	string		A unique configuration ID.
port	int		The port for the JMX/REST connector, typically the HTTPS port.

Configuration Management (config)

Defines how the server processes configuration information.

Attribute name	Data type	Default value	Description
monitorInterval	A period of time with millisecond precision	500ms	Rate at which the server checks for configuration updates. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Action to take after a incurring a configuration error.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

Attribute name	Data type	Default value	Description
updateTrigger	<ul style="list-style-type: none"> • mbean • polled • disabled 	polled	<p>Configuration update method or trigger.</p> <p>mbean Server will only update the configuration when prompted by the FileNotificationMbean. The FileNotificationMbean is typically called by an external program such as an integrated development environment or a management application.</p> <p>polled Server will scan for changes at the polling interval on all the configuration files and update the runtime configuration with the changes detected.</p> <p>disabled Disables all update monitoring. Configuration changes will not be applied while the server is running.</p>

Connection Manager (connectionManager)

Connection Manager configuration

Attribute name	Data type	Default value	Description
agedTimeout	A period of time with second precision	-1	Amount of time before a physical connection can be discarded by pool maintenance. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
connectionTimeout	A period of time with second precision	30s	Amount of time after which a connection request times out. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
id	string		A unique configuration ID.
maxConnectionsPerThread	int Minimum: 0		Limits the number of open connections on each thread.
maxIdleTime	A period of time with second precision	30m	Amount of time after which an unused or idle connection can be discarded during pool maintenance, if doing so does not reduce the pool below the minimum size. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
maxPoolSize	int Minimum: 0	50	Maximum number of physical connections for a pool. A value of 0 means unlimited.
minPoolSize	int Minimum: 0		Minimum number of physical connections to maintain in the pool. The pool is not pre-populated. Aged timeout can override the minimum.
numConnectionsPerThreadLocal	int Minimum: 0		Caches the specified number of connections for each thread.
purgePolicy	<ul style="list-style-type: none"> • ValidateAllConnections • FailingConnectionOnly • EntirePool 	EntirePool	<p>Specifies which connections to destroy when a stale connection is detected in a pool.</p> <p>ValidateAllConnections When a stale connection is detected, connections are tested and those found to be bad are closed.</p> <p>FailingConnectionOnly When a stale connection is detected, only the connection which was found to be bad is closed.</p> <p>EntirePool When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed.</p>
reapTime	A period of time with second precision	3m	Amount of time between runs of the pool maintenance thread. A value of -1 disables pool maintenance. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Constrained Delegation (constrainedDelegation)

Controls the operation of constrained delegation.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
s4U2selfEnabled	boolean	false	Indicate by true or false whether s4U2self is enabled.

Thread Context Propagation (contextService)

Configures how context is propagated to threads

- baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
- classloaderContext
- jeeMetadataContext
- securityContext
- syncToOSThreadContext
- zosWLMContext

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

baseContext > classloaderContext

Classloader context propagation configuration.

false

baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

classloaderContext

Classloader context propagation configuration.

false

jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

Cross-Origin Resource Sharing (cors)

Refers to a cross-domain security procedure for JavaScript-based clients.

Attribute name	Data type	Default value	Description
allowCredentials	boolean		A boolean that indicates if the user credentials can be included in the request.
allowedHeaders	string		A comma-separated list of HTTP headers allowed to be used by the origin domain when making requests to the configured domain.
allowedMethods	string		A comma-separated list of HTTP methods allowed to be used by the origin domain when making requests to the configured domain.
allowedOrigins	string	null	A comma-separated list of origins allowed to access the configured domain.

Attribute name	Data type	Default value	Description
domain	string		The domain name represents the URL being setup with these CORS settings.
exposeHeaders	string		A comma-separated list of HTTP headers that are safe to expose to the calling API.
id	string		A unique configuration ID.
maxAge	long		A long that indicates how many seconds a response to a preflight request can be cached in the browser.

Custom LDAP Filters (customLdapFilterProperties)

Specifies the list of default Custom LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)(objectclass=groupOfDynamicNames)))	An LDAP filter clause for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	An LDAP filter that maps group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

Data Source (dataSource)

Defines a data source configuration.

- connectionManager
- containerAuthData
- jaasLoginContextEntry
- jdbcDriver
 - library
 - file
 - fileset
 - folder
- properties
- properties.datadirect.sqlserver
- properties.db2.i.native

- properties.db2.i.toolbox
- properties.db2.jcc
- properties.derby.client
- properties.derby.embedded
- properties.informix
- properties.informix.jcc
- properties.microsoft.sqlserver
- properties.oracle
- properties.sybase
- recoveryAuthData

Attribute name	Data type	Default value	Description
beginTranForResultSetScrolling	boolean	true	Attempt transaction enlistment when result set scrolling interfaces are used.
beginTranForVendorAPIs	boolean	true	Attempt transaction enlistment when vendor interfaces are used.
commitOrRollbackOnCleanup	<ul style="list-style-type: none"> • commit • rollback 		<p>Determines how to clean up connections that might be in a database unit of work (AutoCommit=false) when the connection is closed or returned to the pool.</p> <p>commit Clean up the connection by committing.</p> <p>rollback Clean up the connection by rolling back.</p>
connectionManagerRef	A reference to top level connectionManager element (string).		Connection manager for a data source.
connectionSharing	<ul style="list-style-type: none"> • MatchOriginalRequest • MatchCurrentState 	MatchOriginalRequest	<p>Specifies how connections are matched for sharing.</p> <p>MatchOriginalRequest When sharing connections, match based on the original connection request.</p> <p>MatchCurrentState When sharing connections, match based on the current state of the connection.</p>

Attribute name	Data type	Default value	Description
containerAuthDataRef	A reference to top level authData element (string).		Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.
enableConnectionCasting	boolean	false	Indicates that connections obtained from the data source should be castable to interface classes that the JDBC vendor connection implementation implements. Enabling this option incurs additional overhead on each getConnection operation. If vendor JDBC interfaces are needed less frequently, it might be more efficient to leave this option disabled and use Connection.unwrap(interface) only where it is needed.
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
isolationLevel	<ul style="list-style-type: none"> • TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT 		<p>Default transaction isolation level.</p> <p>TRANSACTION_REPEATABLE_READ Dirty reads and non-repeatable reads are prevented; phantom reads can occur.</p> <p>TRANSACTION_READ_COMMITTED Dirty reads are prevented; non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SERIALIZABLE Dirty reads, non-repeatable reads and phantom reads are prevented.</p> <p>TRANSACTION_READ_UNCOMMITTED Dirty reads, non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SNAPSHOT Snapshot isolation for Microsoft SQL Server JDBC Driver and DataDirect Connect for JDBC driver.</p>
jaasLoginContextEntryRef	A reference to top level jaasLoginContextEntry element (string).		JAAS login context entry for authentication.
jdbcDriverRef	A reference to top level jdbcDriver element (string).		JDBC driver for a data source.
jndiName	string		JNDI name for a data source.

Attribute name	Data type	Default value	Description
queryTimeout	A period of time with second precision		Default query timeout for SQL statements. In a JTA transaction, syncQueryTimeoutWithTransactionTimeout can override this default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
recoveryAuthDataRef	A reference to top level authData element (string).		Authentication data for transaction recovery.
statementCacheSize	int Minimum: 0	10	Maximum number of cached statements per connection.
supplementalJDBCTrace	boolean		Supplements the JDBC driver trace that is logged when JDBC driver trace is enabled in bootstrap.properties. JDBC driver trace specifications include: com.ibm.ws.database.logwriter, com.ibm.ws.db2.logwriter, com.ibm.ws.derby.logwriter, com.ibm.ws.informix.logwriter, com.ibm.ws.oracle.logwriter, com.ibm.ws.sqlserver.logwriter, com.ibm.ws.sybase.logwriter
syncQueryTimeoutWithTransactionTimeout	boolean	false	Use the time remaining (if any) in a JTA transaction as the default query timeout for SQL statements.
transactional	boolean	true	Enable participation in transactions that are managed by the application server.
type	<ul style="list-style-type: none"> • javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource 		Type of data source. javax.sql.DataSource javax.sql.DataSource javax.sql.XADataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource javax.sql.ConnectionPoolDataSource

connectionManager

Connection manager for a data source.

false

Attribute name	Data type	Default value	Description
agedTimeout	A period of time with second precision	-1	Amount of time before a physical connection can be discarded by pool maintenance. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
connectionTimeout	A period of time with second precision	30s	Amount of time after which a connection request times out. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxConnectionsPerThread	int Minimum: 0		Limits the number of open connections on each thread.
maxIdleTime	A period of time with second precision	30m	Amount of time after which an unused or idle connection can be discarded during pool maintenance, if doing so does not reduce the pool below the minimum size. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxPoolSize	int Minimum: 0	50	Maximum number of physical connections for a pool. A value of 0 means unlimited.

Attribute name	Data type	Default value	Description
minPoolSize	int Minimum: 0		Minimum number of physical connections to maintain in the pool. The pool is not pre-populated. Aged timeout can override the minimum.
numConnectionsPerThreadLocal	int Minimum: 0		Caches the specified number of connections for each thread.
purgePolicy	<ul style="list-style-type: none"> • ValidateAllConnections • FailingConnectionOnly • EntirePool 	EntirePool	<p>Specifies which connections to destroy when a stale connection is detected in a pool.</p> <p>ValidateAllConnections When a stale connection is detected, connections are tested and those found to be bad are closed.</p> <p>FailingConnectionOnly When a stale connection is detected, only the connection which was found to be bad is closed.</p> <p>EntirePool When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed.</p>
reapTime	A period of time with second precision	3m	Amount of time between runs of the pool maintenance thread. A value of -1 disables pool maintenance. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

containerAuthData

Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

jaasLoginContextEntry

JAAS login context entry for authentication.

false

Attribute name	Data type	Default value	Description
loginModuleRef	List of references to top level jaasLoginModule elements (comma-separated string).	hashtable,userNameAndPassword,certificate,clientId	Word, certificate, clientId of a JAAS login module.
name	string		Name of a JAAS configuration entry.

jdbcDriver

JDBC driver for a data source.

false

Attribute name	Data type	Default value	Description
javax.sql.ConnectionPoolDataSource	string		JDBC driver implementation of javax.sql.ConnectionPoolDataSource.
javax.sql.DataSource	string		JDBC driver implementation of javax.sql.DataSource.
javax.sql.XADataSource	string		JDBC driver implementation of javax.sql.XADataSource.
libraryRef	A reference to top level library element (string).		Identifies JDBC driver JARs and native files.

jdbcDriver > library

Identifies JDBC driver JARs and native files.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

jdbcDriver > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

jdbcDriver > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

jdbcDriver > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

properties

List of JDBC vendor properties for the data source. For example, databaseName="dbname" serverName="localhost" portNumber="50000".

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database.
databaseName	string		JDBC driver property: databaseName.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
serverName	string		Server where the database is running.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.datadirect.sqlserver

Data source properties for the DataDirect Connect for JDBC driver for Microsoft SQL Server.

false

Attribute name	Data type	Default value	Description
JDBCBehavior	<ul style="list-style-type: none"> • 1 • 0 	0	JDBC driver property: JDBCBehavior. Values are: 0 (JDBC 4.0) or 1 (JDBC 3.0). 1 JDBC 3.0 0 JDBC 4.0
XATransactionGroup	string		JDBC driver property: XATransactionGroup.
XMLDescribeType	<ul style="list-style-type: none"> • longvarbinary • longvarchar 		JDBC driver property: XMLDescribeType. longvarbinary longvarbinary longvarchar longvarchar
accountingInfo	string		JDBC driver property: accountingInfo.
alternateServers	string		JDBC driver property: alternateServers.
alwaysReportTriggerResults	boolean		JDBC driver property: alwaysReportTriggerResults.
applicationName	string		JDBC driver property: applicationName.
authenticationMethod	<ul style="list-style-type: none"> • ntlm • userIdPassword • kerberos • auto 		JDBC driver property: authenticationMethod. ntlm ntlm userIdPassword userIdPassword kerberos kerberos auto auto
bulkLoadBatchSize	long		JDBC driver property: bulkLoadBatchSize.
bulkLoadOptions	long		JDBC driver property: bulkLoadOptions.
clientHostName	string		JDBC driver property: clientHostName.
clientUser	string		JDBC driver property: clientUser.

Attribute name	Data type	Default value	Description
codePageOverride	string		JDBC driver property: codePageOverride.
connectionRetryCount	int		JDBC driver property: connectionRetryCount.
connectionRetryDelay	A period of time with second precision		JDBC driver property: connectionRetryDelay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
convertNull	int		JDBC driver property: convertNull.
databaseName	string		JDBC driver property: databaseName.
dateTimeInputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeInputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
dateTimeOutputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeOutputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
describeInputParameters	<ul style="list-style-type: none"> • describeIfString • noDescribe • describeIfDateTime • describeAll 		JDBC driver property: describeInputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll

Attribute name	Data type	Default value	Description
describeOutputParameters	<ul style="list-style-type: none"> describeIfString noDescribe describeIfDateTime describeAll 		JDBC driver property: describeOutputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll
enableBulkLoad	boolean		JDBC driver property: enableBulkLoad.
enableCancelTimeout	boolean		JDBC driver property: enableCancelTimeout.
encryptionMethod	<ul style="list-style-type: none"> loginSSL requestSSL SSL noEncryption 		JDBC driver property: encryptionMethod. loginSSL loginSSL requestSSL requestSSL SSL SSL noEncryption noEncryption
failoverGranularity	<ul style="list-style-type: none"> disableIntegrityCheck atomicWithRepositioning nonAtomic atomic 		JDBC driver property: failoverGranularity. disableIntegrityCheck disableIntegrityCheck atomicWithRepositioning atomicWithRepositioning nonAtomic nonAtomic atomic atomic
failoverMode	<ul style="list-style-type: none"> connect select extended 		JDBC driver property: failoverMode. connect connect select select extended extended
failoverPreconnect	boolean		JDBC driver property: failoverPreconnect.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
initializationString	string		JDBC driver property: initializationString.
insensitiveResultSetBufferSize	int		JDBC driver property: insensitiveResultSetBufferSize.

Attribute name	Data type	Default value	Description
javaDoubleToString	boolean		JDBC driver property: javaDoubleToString.
loadBalancing	boolean		JDBC driver property: loadBalancing.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
longDataCacheSize	int Minimum: -1		JDBC driver property: longDataCacheSize.
netAddress	string		JDBC driver property: netAddress.
packetSize	int Minimum: -1 Maximum: 128		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
queryTimeout	A period of time with second precision		JDBC driver property: queryTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
resultsetMetaDataOptions	int		JDBC driver property: resultSetMetaDataOptions.
selectMethod	<ul style="list-style-type: none"> • direct • cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
serverName	string	localhost	Server where the database is running.
snapshotSerializable	boolean		JDBC driver property: snapshotSerializable.

Attribute name	Data type	Default value	Description
spyAttributes	string		JDBC driver property: spyAttributes.
stringInputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringInputParameterType. varchar varchar nvarchar nvarchar
stringOutputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringOutputParameterType. varchar varchar nvarchar nvarchar
suppressConnectionWarnings	boolean		JDBC driver property: suppressConnectionWarnings.
transactionMode	<ul style="list-style-type: none"> explicit implicit 		JDBC driver property: transactionMode. explicit explicit implicit implicit
truncateFractionalSeconds	boolean		JDBC driver property: truncateFractionalSeconds.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
useServerSideUpdatableCursors	boolean		JDBC driver property: useServerSideUpdatableCursors.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
validateServerCertificate	boolean		JDBC driver property: validateServerCertificate.

properties.db2.i.native

Data source properties for the IBM DB2 for i Native JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> read only all read call 	all	JDBC driver property: access. read only read only all all read call read call

Attribute name	Data type	Default value	Description
autoCommit	boolean	true	JDBC driver property: autoCommit.
batchStyle	<ul style="list-style-type: none"> • 2.1 • 2.0 	2.0	JDBC driver property: batchStyle. 2.1 2.1 2.0 2.0
behaviorOverride	int		JDBC driver property: behaviorOverride.
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive
dataTruncation	string	true	JDBC driver property: dataTruncation.
databaseName	string	*LOCAL	JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy
dateSeparator	<ul style="list-style-type: none"> • \, • b • . • / • - 		JDBC driver property: dateSeparator. \, The comma character (,). b The character b . The period character (.). / The forward slash character (/). - The dash character (-).
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		JDBC driver property: decimalSeparator. \, The comma character (,). . The period character (.).
directMap	boolean	true	JDBC driver property: directMap.
doEscapeProcessing	boolean	true	JDBC driver property: doEscapeProcessing.
fullErrors	boolean		JDBC driver property: fullErrors.
libraries	string		JDBC driver property: libraries.
lobThreshold	int Maximum: 500000	0	JDBC driver property: lobThreshold.

Attribute name	Data type	Default value	Description
lockTimeout	A period of time with second precision	0	JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 63
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
networkProtocol	int		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
prefetch	boolean	true	JDBC driver property: prefetch.
queryOptimizeGoal	<ul style="list-style-type: none"> • 2 • 1 	2	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO). 2 *ALLIO 1 *FIRSTIO

Attribute name	Data type	Default value	Description
reuseObjects	boolean	true	JDBC driver property: reuseObjects.
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
systemNaming	boolean	false	JDBC driver property: systemNaming.
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • \, • b • : • . 		JDBC driver property: timeSeparator. \, The comma character (,). b The character b : The colon character (:). . The period character (.).
trace	boolean		JDBC driver property: trace.
transactionTimeout	A period of time with second precision	0	JDBC driver property: transactionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateHex	<ul style="list-style-type: none"> • binary • character 	character	JDBC driver property: translateHex. binary binary character character
useBlockInsert	boolean	false	JDBC driver property: useBlockInsert.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.db2.i.toolbox

Data source properties for the IBM DB2 for i Toolbox JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> • read only • all • read call 	all	JDBC driver property: access. read only read only all all read call read call
behaviorOverride	int		JDBC driver property: behaviorOverride.
bidiImplicitReordering	boolean	true	JDBC driver property: bidiImplicitReordering.
bidiNumericOrdering	boolean	false	JDBC driver property: bidiNumericOrdering.
bidiStringType	int		JDBC driver property: bidiStringType.
bigDecimal	boolean	true	JDBC driver property: bigDecimal.
blockCriteria	<ul style="list-style-type: none"> • 2 • 1 • 0 	2	JDBC driver property: blockCriteria. Values are: 0 (no record blocking), 1 (block if FOR FETCH ONLY is specified), 2 (block if FOR UPDATE is specified). 2 2 1 1 0 0

Attribute name	Data type	Default value	Description
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive • insensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive insensitive insensitive
dataCompression	boolean	true	JDBC driver property: dataCompression.
dataTruncation	boolean	true	JDBC driver property: dataTruncation.
databaseName	string		JDBC driver property: databaseName.
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy

Attribute name	Data type	Default value	Description
dateSeparator	<ul style="list-style-type: none"> • • \, • . • / • - 		<p>JDBC driver property: dateSeparator.</p> <p>The space character ().</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p> <p>/ The forward slash character (/).</p> <p>- The dash character (-).</p>
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		<p>JDBC driver property: decimalSeparator.</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p>
driver	<ul style="list-style-type: none"> • toolbox • native 	toolbox	<p>JDBC driver property: driver.</p> <p>toolbox toolbox</p> <p>native native</p>
errors	<ul style="list-style-type: none"> • full • basic 	basic	<p>JDBC driver property: errors.</p> <p>full full</p> <p>basic basic</p>
extendedDynamic	boolean	false	JDBC driver property: extendedDynamic.
extendedMetaData	boolean	false	JDBC driver property: extendedMetaData.
fullOpen	boolean	false	JDBC driver property: fullOpen.
holdInputLocators	boolean	true	JDBC driver property: holdInputLocators.
holdStatements	boolean	false	JDBC driver property: holdStatements.
isolationLevelSwitchingSupport	boolean	false	JDBC driver property: isolationLevelSwitchingSupport.
keepAlive	boolean		JDBC driver property: keepAlive.
lazyClose	boolean	false	JDBC driver property: lazyClose.
libraries	string		JDBC driver property: libraries.

Attribute name	Data type	Default value	Description
lobThreshold	int Minimum: 0 Maximum: 16777216	0	JDBC driver property: lobThreshold.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 64
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
metaDataSource	int Minimum: 0 Maximum: 1	1	JDBC driver property: metaDataSource.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
naming	<ul style="list-style-type: none"> • system • sql 	sql	JDBC driver property: naming. system system sql sql
package	string		JDBC driver property: package.
packageAdd	boolean	true	JDBC driver property: packageAdd.
packageCCSID	<ul style="list-style-type: none"> • 13488 • 1200 	13488	JDBC driver property: packageCCSID. Values are: 1200 (UCS-2) or 13488 (UTF-16). 13488 13488 (UTF-16) 1200 1200 (UCS-2)
packageCache	boolean	false	JDBC driver property: packageCache.

Attribute name	Data type	Default value	Description
packageCriteria	<ul style="list-style-type: none"> default select 	default	JDBC driver property: packageCriteria. default default select select
packageError	<ul style="list-style-type: none"> exception none warning 	warning	JDBC driver property: packageError. exception exception none none warning warning
packageLibrary	string	QGPL	JDBC driver property: packageLibrary.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
prefetch	boolean	true	JDBC driver property: prefetch.
prompt	boolean	false	JDBC driver property: prompt.
proxyServer	string		JDBC driver property: proxyServer.
qaqqiniLibrary	string		JDBC driver property: qaqqiniLibrary.
queryOptimizeGoal	int Minimum: 0 Maximum: 2	0	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO).
receiveBufferSize	int Minimum: 1		JDBC driver property: receiveBufferSize.
remarks	<ul style="list-style-type: none"> system sql 	system	JDBC driver property: remarks. system system sql sql
rollbackCursorHold	boolean	false	JDBC driver property: rollbackCursorHold.
savePasswordWhenSerialized	boolean	false	JDBC driver property: savePasswordWhenSerialized.
secondaryUrl	string		JDBC driver property: secondaryUrl.
secure	boolean	false	JDBC driver property: secure.
sendBufferSize	int Minimum: 1		JDBC driver property: sendBufferSize.

Attribute name	Data type	Default value	Description
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
soLinger	A period of time with second precision		JDBC driver property: soLinger. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
soTimeout	A period of time with millisecond precision		JDBC driver property: soTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
sort	<ul style="list-style-type: none"> • hex • table • language 	hex	JDBC driver property: sort. hex hex table table language language
sortLanguage	string		JDBC driver property: sortLanguage.
sortTable	string		JDBC driver property: sortTable.
sortWeight	<ul style="list-style-type: none"> • unqiue • shared 		JDBC driver property: sortWeight. unqiue unique shared shared
tcpNoDelay	boolean		JDBC driver property: tcpNoDelay.
threadUsed	boolean	true	JDBC driver property: threadUsed.

Attribute name	Data type	Default value	Description
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • • \, • : • . 		JDBC driver property: timeSeparator. The space character (). \, The comma character (,). : The colon character (:). . The period character (.).
toolboxTrace	<ul style="list-style-type: none"> • diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning 		JDBC driver property: toolboxTrace. diagnostic diagnostic information information conversion conversion error error thread thread proxy proxy none none datastream datastream pcml pcml all all jdbc jdbc warning warning
trace	boolean		JDBC driver property: trace.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateBoolean	boolean	true	JDBC driver property: translateBoolean.

Attribute name	Data type	Default value	Description
translateHex	<ul style="list-style-type: none"> binary character 	character	JDBC driver property: translateHex. binary binary character character
trueAutoCommit	boolean	false	JDBC driver property: trueAutoCommit.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaLooselyCoupledSupport	int Minimum: 0 Maximum: 1	0	JDBC driver property: xaLooselyCoupledSupport.

properties.db2.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for DB2.

false

Attribute name	Data type	Default value	Description
activateDatabase	int		JDBC driver property: activateDatabase.
alternateGroupDatabaseName	string		JDBC driver property: alternateGroupDatabaseName.
alternateGroupPortNumber	string		JDBC driver property: alternateGroupPortNumber.
alternateGroupServerName	string		JDBC driver property: alternateGroupServerName.
blockingReadConnectionTimeout	A period of time with second precision		JDBC driver property: blockingReadConnectionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
clientAccountingInformation	string		JDBC driver property: clientAccountingInformation.
clientApplicationInformation	string		JDBC driver property: clientApplicationInformation.
clientRerouteAlternatePortNumber	string		JDBC driver property: clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	string		JDBC driver property: clientRerouteAlternateServerName.

Attribute name	Data type	Default value	Description
clientUser	string		JDBC driver property: clientUser.
clientWorkstation	string		JDBC driver property: clientWorkstation.
connectionCloseWithInFlightTransaction	Transaction • 1		JDBC driver property: connectionCloseWithInFlightTransaction. 2 CONNECTION_CLOSE_WITH_ROLLBACK 1 CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	int		JDBC driver property: currentAlternateGroupEntry.
currentFunctionPath	string		JDBC driver property: currentFunctionPath.
currentLocaleLcCtype	string		JDBC driver property: currentLocaleLcCtype.
currentLockTimeout	A period of time with second precision		JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
currentPackagePath	string		JDBC driver property: currentPackagePath.
currentPackageSet	string		JDBC driver property: currentPackageSet.
currentSQLID	string		JDBC driver property: currentSQLID.
currentSchema	string		JDBC driver property: currentSchema.
cursorSensitivity	• 2 • 1 • 0		JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). 2 TYPE_SCROLL_ASENSITIVE 1 TYPE_SCROLL_SENSITIVE_DYNAMIC 0 TYPE_SCROLL_SENSITIVE_STATIC
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean	true	JDBC driver property: deferPrepares.

Attribute name	Data type	Default value	Description
driverType	<ul style="list-style-type: none"> 2 4 	4	JDBC driver property: driverType. 2 Type 2 JDBC driver. 4 Type 4 JDBC driver.
enableAlternateGroupSeamlessACR	boolean		JDBC driver property: enableAlternateGroupSeamlessACR.
enableClientAffinitiesList	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableClientAffinitiesList. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableExtendedDescribe	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedDescribe. 2 NO 1 YES
enableExtendedIndicators	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedIndicators. 2 NO 1 YES
enableNamedParameterMarkers	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSeamlessFailover	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeInputStreams	boolean		JDBC driver property: fullyMaterializeInputStreams.
fullyMaterializeInputStreamsOnBatchExecution	<ul style="list-style-type: none"> 1 		JDBC driver property: fullyMaterializeInputStreamsOnBatchExecution. 2 NO 1 YES
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.

Attribute name	Data type	Default value	Description
implicitRollbackOption	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: implicitRollbackOption.</p> <p>2 IMPLICIT_ROLLBACK_OPTION_N</p> <p>1 IMPLICIT_ROLLBACK_OPTION_N</p> <p>0 IMPLICIT_ROLLBACK_OPTION_N</p>
interruptProcessingMode	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: interruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODI</p> <p>1 INTERRUPT_PROCESSING_MODI</p> <p>0 INTERRUPT_PROCESSING_MODI</p>
keepAliveTimeOut	A period of time with second precision		<p>JDBC driver property: keepAliveTimeOut. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
keepDynamic	int		JDBC driver property: keepDynamic.
kerberosServerPrincipal	string		JDBC driver property: kerberosServerPrincipal.
loginTimeout	A period of time with second precision		<p>JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
maxConnCachedParamBufferSize	int		JDBC driver property: maxConnCachedParamBufferSize.
maxRetriesForClientReroute	int		JDBC driver property: maxRetriesForClientReroute.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	50000	Port on which to obtain database connections.
profileName	string		JDBC driver property: profileName.

Attribute name	Data type	Default value	Description
queryCloseImplicit	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryCloseImplicit. Values are: 1 (QUERY_CLOSE_IMPLICIT_YES) or 2 (QUERY_CLOSE_IMPLICIT_NO).</p> <p>2 QUERY_CLOSE_IMPLICIT_NO 1 QUERY_CLOSE_IMPLICIT_YES</p>
queryDataSize	<p>int</p> <p>Minimum: 4096</p> <p>Maximum: 65535</p>		JDBC driver property: queryDataSize.
queryTimeoutInterruptProcessingMode	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryTimeoutInterruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODE_C 1 INTERRUPT_PROCESSING_MODE_S</p>
readOnly	boolean		JDBC driver property: readOnly.
recordTemporalHistory	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: recordTemporalHistory.</p> <p>2 NO 1 YES</p>
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.

Attribute name	Data type	Default value	Description
retryIntervalForClientReroute	A period of time with second precision		JDBC driver property: retryIntervalForClientReroute. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12 		<p>JDBC driver property: securityMechanism. Values are:</p> <ul style="list-style-type: none"> 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY), 11 (KERBEROS_SECURITY), 12 (ENCRYPTED_USER_AND_DATA_SECURITY), 13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY), 15 (PLUGIN_SECURITY), 16 (ENCRYPTED_USER_ONLY_SECURITY), 18 (TLS_CLIENT_CERTIFICATE_SECURITY). <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 18 TLS_CLIENT_CERTIFICATE_SECURITY 15 PLUGIN_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY 16 ENCRYPTED_USER_ONLY_SECURITY 13 ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY 11 KERBEROS_SECURITY 12 ENCRYPTED_USER_AND_DATA_SECURITY</p>
sendDataAsIs	boolean		JDBC driver property: sendDataAsIs.
serverName	string	localhost	Server where the database is running.
sessionTimeZone	string		JDBC driver property: sessionTimeZone.
sqljCloseStmtsWithOpenResultSet	boolean		JDBC driver property: sqljCloseStmtsWithOpenResultSet.

Attribute name	Data type	Default value	Description
sqljEnableClassLoaderSpecificProfiles	boolean		JDBC driver property: sqljEnableClassLoaderSpecificProfiles.
sslConnection	boolean		JDBC driver property: sslConnection.
streamBufferSize	int		JDBC driver property: streamBufferSize.
stripTrailingZerosForDecimalNumbers	Numbers <ul style="list-style-type: none"> 1 		JDBC driver property: stripTrailingZerosForDecimalNumbers. 2 NO 1 YES
sysSchema	string		JDBC driver property: sysSchema.
timerLevelForQueryTimeOut	<ul style="list-style-type: none"> 2 1 -1 		JDBC driver property: timerLevelForQueryTimeOut. 2 QUERYTIMEOUT_CONNECTION_LEVEL 1 QUERYTIMEOUT_STATEMENT_LEVEL -1 QUERYTIMEOUT_DISABLED
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceFileCount	int		JDBC driver property: traceFileCount.
traceFileSize	int		JDBC driver property: traceFileSize.
traceLevel	int	0	Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
traceOption	<ul style="list-style-type: none"> 1 0 		JDBC driver property: traceOption 1 1 0 0

Attribute name	Data type	Default value	Description
translateForBitData	<ul style="list-style-type: none"> 2 1 		JDBC driver property: translateForBitData. 2 SERVER_ENCODING_REPRESENTATION 1 HEX_REPRESENTATION
updateCountForBatch	<ul style="list-style-type: none"> 2 1 		JDBC driver property: updateCountForBatch. 2 TOTAL_UPDATE_COUNT 1 NO_UPDATE_COUNT
useCachedCursor	boolean		JDBC driver property: useCachedCursor.
useIdentityValLocalForAutoGeneratedKeys	boolean		JDBC driver property: useIdentityValLocalForAutoGeneratedKeys.
useJDBC41DefinitionForGetColumns	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC41DefinitionForGetColumns. 2 NO 1 YES
useJDBC4ColumnNameAndLabelSemantics	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
useTransactionRedirect	boolean		JDBC driver property: useTransactionRedirect.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaNetworkOptimization	boolean		JDBC driver property: xaNetworkOptimization.

properties.derby.client

Data source properties for Derby Network Client JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1527	Port on which to obtain database connections.
retrieveMessageText	boolean	true	JDBC driver property: retrieveMessageText.
securityMechanism	<ul style="list-style-type: none"> 3 7 4 9 8 	3	<p>JDBC driver property: securityMechanism. Values are:</p> <p>3 (CLEAR_TEXT_PASSWORD_SECURITY),</p> <p>4 (USER_ONLY_SECURITY),</p> <p>7 (ENCRYPTED_PASSWORD_SECURITY),</p> <p>8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY),</p> <p>9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY</p> <p>7 ENCRYPTED_PASSWORD_SECURITY</p> <p>4 USER_ONLY_SECURITY</p> <p>9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p> <p>8 STRONG_PASSWORD_SUBSTITUTE_SECURITY</p>

Attribute name	Data type	Default value	Description
serverName	string	localhost	Server where the database is running.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		JDBC driver property: shutdownDatabase. false Do not shut down the database. shutdown Shut down the database when a connection is attempted.
ssl	<ul style="list-style-type: none"> basic off peerAuthentication 		JDBC driver property: ssl. basic basic off off peerAuthentication peerAuthentication
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_XA_CALLS=2048, TRACE_ALL=-1.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.derby.embedded

Data source properties for Derby Embedded JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		<p>JDBC driver property: shutdownDatabase.</p> <p>false Do not shut down the database.</p> <p>shutdown Shut down the database when a connection is attempted.</p>
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.informix

Data source properties for the Informix JDBC driver.

false

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
ifxCLIENT_LOCALE	string		JDBC driver property: ifxCLIENT_LOCALE.
ifxCPMAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMInitPoolSize	int		JDBC driver property: ifxCPMInitPoolSize.
ifxCPMMaxConnections	int		JDBC driver property: ifxCPMMaxConnections.
ifxCPMMaxPoolSize	int		JDBC driver property: ifxCPMMaxPoolSize.
ifxCPMMinAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMMinAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMMinPoolSize	int		JDBC driver property: ifxCPMMinPoolSize.
ifxCPMServiceInterval	A period of time with millisecond precision		JDBC driver property: ifxCPMServiceInterval. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxDBANSIWARN	boolean		JDBC driver property: ifxDBANSIWARN.
ifxDBCENTURY	string		JDBC driver property: ifxDBCENTURY.
ifxDBDATE	string		JDBC driver property: ifxDBDATE.
ifxDBSPACETEMP	string		JDBC driver property: ifxDBSPACETEMP.

Attribute name	Data type	Default value	Description
ifxDBTEMP	string		JDBC driver property: ifxDBTEMP.
ifxDBTIME	string		JDBC driver property: ifxDBTIME.
ifxDBUPSPACE	string		JDBC driver property: ifxDBUPSPACE.
ifxDB_LOCALE	string		JDBC driver property: ifxDB_LOCALE.
ifxDELIMIDENT	boolean		JDBC driver property: ifxDELIMIDENT.
ifxENABLE_TYPE_CACHE	boolean		JDBC driver property: ifxENABLE_TYPE_CACHE.
ifxFET_BUF_SIZE	int		JDBC driver property: ifxFET_BUF_SIZE.
ifxGL_DATE	string		JDBC driver property: ifxGL_DATE.
ifxGL_DATETIME	string		JDBC driver property: ifxGL_DATETIME.
ifxIFXHOST	string	localhost	JDBC driver property: ifxIFXHOST.
ifxIFX_AUTOFREE	boolean		JDBC driver property: ifxIFX_AUTOFREE.
ifxIFX_DIRECTIVES	string		JDBC driver property: ifxIFX_DIRECTIVES.
ifxIFX_LOCK_MODE_WAIT	A period of time with second precision	2s	JDBC driver property: ifxIFX_LOCK_MODE_WAIT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxIFX_SOC_TIMEOUT	A period of time with millisecond precision		JDBC driver property: ifxIFX_SOC_TIMEOUT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxIFX_USEPUT	boolean		JDBC driver property: ifxIFX_USEPUT.

Attribute name	Data type	Default value	Description
ifxIFX_USE_STRENC	boolean		JDBC driver property: ifxIFX_USE_STRENC.
ifxIFX_XASPEC	string	y	JDBC driver property: ifxIFX_XASPEC.
ifxINFORMIXCONRETRY	int		JDBC driver property: ifxINFORMIXCONRETRY.
ifxINFORMIXCONTIME	A period of time with second precision		JDBC driver property: ifxINFORMIXCONTIME. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxINFORMIXOPCACHE	string		JDBC driver property: ifxINFORMIXOPCACHE.
ifxINFORMIXSTACKSIZE	int		JDBC driver property: ifxINFORMIXSTACKSIZE.
ifxJDBCTEMP	string		JDBC driver property: ifxJDBCTEMP.
ifxLDAP_IFXBASE	string		JDBC driver property: ifxLDAP_IFXBASE.
ifxLDAP_PASSWD	string		JDBC driver property: ifxLDAP_PASSWD.
ifxLDAP_URL	string		JDBC driver property: ifxLDAP_URL.
ifxLDAP_USER	string		JDBC driver property: ifxLDAP_USER.
ifxLOBCACHE	int		JDBC driver property: ifxLOBCACHE.
ifxNEWCODESET	string		JDBC driver property: ifxNEWCODESET.
ifxNEWLOCALE	string		JDBC driver property: ifxNEWLOCALE.
ifxNODEFDAC	string		JDBC driver property: ifxNODEFDAC.
ifxOPTCOMPIND	string		JDBC driver property: ifxOPTCOMPIND.
ifxOPTOFC	string		JDBC driver property: ifxOPTOFC.
ifxOPT_GOAL	string		JDBC driver property: ifxOPT_GOAL.
ifxPATH	string		JDBC driver property: ifxPATH.
ifxPDQPRIORITY	string		JDBC driver property: ifxPDQPRIORITY.

Attribute name	Data type	Default value	Description
ifxPLCONFIG	string		JDBC driver property: ifxPLCONFIG.
ifxPLOAD_LO_PATH	string		JDBC driver property: ifxPLOAD_LO_PATH.
ifxPROTOCOLTRACE	int		JDBC driver property: ifxPROTOCOLTRACE.
ifxPROTOCOLTRACEFILE	string		JDBC driver property: ifxPROTOCOLTRACEFILE.
ifxPROXY	string		JDBC driver property: ifxPROXY.
ifxPSORT_DBTEMP	string		JDBC driver property: ifxPSORT_DBTEMP.
ifxPSORT_NPROCS	boolean		JDBC driver property: ifxPSORT_NPROCS.
ifxSECURITY	string		JDBC driver property: ifxSECURITY.
ifxSQLH_FILE	string		JDBC driver property: ifxSQLH_FILE.
ifxSQLH_LOC	string		JDBC driver property: ifxSQLH_LOC.
ifxSQLH_TYPE	string		JDBC driver property: ifxSQLH_TYPE.
ifxSSLCONNECTION	string		JDBC driver property: ifxSSLCONNECTION.
ifxSTMT_CACHE	string		JDBC driver property: ifxSTMT_CACHE.
ifxTRACE	int		JDBC driver property: ifxTRACE.
ifxTRACEFILE	string		JDBC driver property: ifxTRACEFILE.
ifxTRUSTED_CONTEXT	string		JDBC driver property: ifxTRUSTED_CONTEXT.
ifxUSEV5SERVER	boolean		JDBC driver property: ifxUSEV5SERVER.
ifxUSE_DTENV	boolean		JDBC driver property: ifxUSE_DTENV.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
roleName	string		JDBC driver property: roleName.
serverName	string		Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.informix.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for Informix.

false

Attribute name	Data type	Default value	Description
DBANSIWARN	boolean		JDBC driver property: DBANSIWARN.
DBDATE	string		JDBC driver property: DBDATE.
DBPATH	string		JDBC driver property: DBPATH.
DBSPACETEMP	string		JDBC driver property: DBSPACETEMP.
DBTEMP	string		JDBC driver property: DBTEMP.
DBUPSPACE	string		JDBC driver property: DBUPSPACE.
DELIMIDENT	boolean		JDBC driver property: DELIMIDENT.
IFX_DIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_DIRECTIVES. ON ON OFF OFF
IFX_EXTDIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_EXTDIRECTIVES. ON ON OFF OFF
IFX_UPDESC	string		JDBC driver property: IFX_UPDESC.

Attribute name	Data type	Default value	Description
IFX_XASTDCOMPLIANCE_XAEND	<ul style="list-style-type: none"> 0 		JDBC driver property: IFX_XASTDCOMPLIANCE_XAEND. 1 1 0 0
INFORMIXOPCACHE	string		JDBC driver property: INFORMIXOPCACHE.
INFORMIXSTACKSIZE	string		JDBC driver property: INFORMIXSTACKSIZE.
NODEFDAC	<ul style="list-style-type: none"> yes no 		JDBC driver property: NODEFDAC. yes yes no no
OPTCOMPIND	<ul style="list-style-type: none"> 2 1 0 		JDBC driver property: OPTCOMPIND. 2 2 1 1 0 0
OPTOFC	<ul style="list-style-type: none"> 1 0 		JDBC driver property: OPTOFC. 1 1 0 0
PDQPRIORITY	<ul style="list-style-type: none"> HIGH LOW OFF 		JDBC driver property: PDQPRIORITY. HIGH HIGH LOW LOW OFF OFF
PSORT_DBTEMP	string		JDBC driver property: PSORT_DBTEMP.
PSORT_NPROCS	string Maximum: 10		JDBC driver property: PSORT_NPROCS.
STMT_CACHE	<ul style="list-style-type: none"> 1 0 		JDBC driver property: STMT_CACHE. 1 1 0 0
currentLockTimeout	A period of time with second precision	2s	JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean		JDBC driver property: deferPrepares.
driverType	int	4	JDBC driver property: driverType.
enableNamedParameterMarkers	int		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO).
enableSeamlessFailover	int		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO).
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.
keepDynamic	int		JDBC driver property: keepDynamic.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
progressiveStreaming	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: progressiveStreaming. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
queryDataSize	int Minimum: 4096 Maximum: 10485760		JDBC driver property: queryDataSize.

Attribute name	Data type	Default value	Description
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 9 		<p>JDBC driver property: securityMechanism. Values are: 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY).</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p>
serverName	string	localhost	Server where the database is running.
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.

Attribute name	Data type	Default value	Description
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
useJDBC4ColumnNameAndLabelSemantics	boolean		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO).
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.microsoft.sqlserver

Data source properties for Microsoft SQL Server JDBC Driver.

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database. Example: jdbc:sqlserver://localhost:1433;databaseName=myDB.
applicationIntent	<ul style="list-style-type: none"> ReadOnly ReadWrite 		JDBC driver property: applicationIntent. ReadOnly ReadOnly ReadWrite ReadWrite
applicationName	string		JDBC driver property: applicationName.
authenticationScheme	<ul style="list-style-type: none"> NativeAuthentication JavaKerberos 		JDBC driver property: authenticationScheme. NativeAuthentication NativeAuthentication JavaKerberos JavaKerberos

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
encrypt	boolean		JDBC driver property: encrypt.
failoverPartner	string		JDBC driver property: failoverPartner.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
instanceName	string		JDBC driver property: instanceName.
integratedSecurity	boolean		JDBC driver property: integratedSecurity.
lastUpdateCount	boolean		JDBC driver property: lastUpdateCount.
lockTimeout	A period of time with millisecond precision		JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
multiSubnetFailover	boolean		JDBC driver property: multiSubnetFailover.
packetSize	int Minimum: 512 Maximum: 32767		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.

Attribute name	Data type	Default value	Description
responseBuffering	<ul style="list-style-type: none"> full adaptive 		JDBC driver property: responseBuffering. full full adaptive adaptive
selectMethod	<ul style="list-style-type: none"> direct cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
sendStringParametersAsUnicode	boolean	false	JDBC driver property: sendStringParametersAsUnicode.
sendTimeAsDatetime	boolean		JDBC driver property: sendTimeAsDatetime.
serverName	string	localhost	Server where the database is running.
trustServerCertificate	boolean		JDBC driver property: trustServerCertificate.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
workstationID	string		JDBC driver property: workstationID.
xopenStates	boolean		JDBC driver property: xopenStates.

properties.oracle

Data source properties for Oracle JDBC driver.

false

Attribute name	Data type	Default value	Description
ONSConfiguration	string		JDBC driver property: ONSConfiguration.
TNSEntryName	string		JDBC driver property: TNSEntryName.
URL	string		URL for connecting to the database. Examples: jdbc:oracle:thin://localhost:1521/sample or jdbc:oracle:oci://localhost:1521/sample.
connectionProperties	string		JDBC driver property: connectionProperties.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
driverType	<ul style="list-style-type: none"> oci thin 	thin	JDBC driver property: driverType. oci oci thin thin
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	string		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1521	Port on which to obtain database connections.
serverName	string	localhost	Server where the database is running.
serviceName	string		JDBC driver property: serviceName.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

properties.sybase

Data source properties for Sybase JDBC driver.

false

Attribute name	Data type	Default value	Description
SERVER_INITIATED_TRANSACTIONS	<ul style="list-style-type: none"> false true 	false	JDBC driver property: SERVER_INITIATED_TRANSACTIONS. false false true true
connectionProperties	string	SELECT_OPENS_CURSOR=true	JDBC driver property: connectionProperties.
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	<ul style="list-style-type: none"> • SSL • socket 		JDBC driver property: networkProtocol. SSL SSL socket socket
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	5000	Port on which to obtain database connections.
resourceManagerName	string		JDBC driver property: resourceManagerName.
serverName	string	localhost	Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
version	int		JDBC driver property: version.

recoveryAuthData

Authentication data for transaction recovery.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

Distributed Map (distributedMap)

Distributed map configuration for a local cache.

- cacheGroup
 - member
 - adapterBeanName
- diskCache
- library
 - file
 - fileset
 - folder

Attribute name	Data type	Default value	Description
cacheProviderName	string	default	Specifies the name of an alternate cache provider.
highThreshold	int Minimum: -1 Maximum: 100	-1	Specifies when the memory cache eviction policy starts. The threshold is expressed in terms of the percentage of the memory cache size in megabytes (MB).
id	string		A unique configuration ID.
jndiName	string	\${id}	JNDI name for a cache instance.
libraryRef	A reference to top level library element (string).		Specifies a reference to a shared library.
lowThreshold	int Minimum: -1 Maximum: 100	-1	Specifies when the memory cache eviction policy ends. The threshold is expressed in terms of the percentage of the memory cache size in megabytes (MB).
memorySizeInEntries	int Minimum: 0	2000	Specifies a positive integer that defines the maximum number of entries that the cache can hold. Values are usually in the thousands. The minimum value is 100, with no set maximum value. The default value is 2000.
memorySizeInMB	int Minimum: -1	-1	Specifies a value for the maximum memory cache size in megabytes (MB).

cacheGroup

Specifies sets of external caches that are controlled by WebSphere(R) Application Server on servers such as IBM(R) WebSphere(R) Edge Server and IBM(R) HTTP Server.

false

Attribute name	Data type	Default value	Description
name	string		Specifies a unique name for the external cache group. The external cache group name must match the ExternalCache property that is defined in the servlet or Java(TM) Server Pages (JSP) cachespec.xml file.

cacheGroup > member

Members of an external cache group that are controlled by WebSphere Application Server.

false

Attribute name	Data type	Default value	Description
host	string		Fully qualified host name
port	int Minimum: 0		Port.

cacheGroup > member > adapterBeanName

Specifies the name of a class, which is located on the WebSphere Application Server class path, of the adapter between WebSphere Application Server and this external cache.

false

string

diskCache

Enable disk offload to specify that when the cache is full, cache entries are removed from the cache and saved to disk. The location is a fully-qualified directory location that is used by the disk offload function. The Flush to Disk on Stop option specifies that when the server stops, the contents of the memory cache are moved to disk.

false

Attribute name	Data type	Default value	Description
evictionPolicy	<ul style="list-style-type: none"> • RANDOM • SIZE 	RANDOM	<p>Specifies the eviction algorithm and thresholds that the disk cache uses to evict entries. When the disk size reaches a high threshold limit, the disk cache garbage collector wakes up and evicts randomly-selected (Random) or the largest (Size) entries on the disk until the disk size reaches a low threshold limit.</p> <p>RANDOM Random</p> <p>SIZE Size</p>

Attribute name	Data type	Default value	Description
flushToDiskOnStopEnabled	boolean	false	Set this value to true to have objects that are cached in memory saved to disk when the server stops. This value is ignored if Enable disk offload is set to false.
highThreshold	int Minimum: 0 Maximum: 100	80	Specifies when the eviction policy starts.
location	Path to a directory		Specifies a directory to use for disk offload.
lowThreshold	int Minimum: 0 Maximum: 100	70	Specifies when the eviction policy ends.
sizeInEntries	int Minimum: 0	100000	Specifies a value for the maximum disk cache size, in number of entries.
sizeInGB	int Minimum: 3	3	Specifies a value for the maximum disk cache size, in gigabytes (GB).

library

Specifies a reference to a shared library.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

IBM Lotus Domino LDAP Filters (domino50LdapFilterProperties)

Specifies the list of default IBM Lotus Domino LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=dominoGroup))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	dominoGroup:member	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=Person))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	person:uid	An LDAP filter that maps the name of a user to an LDAP entry.

Novell eDirectory LDAP Filters (edirectoryLdapFilterProperties)

Specifies the list of Novell eDirectory LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=groupOfNames))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	groupOfNames:member	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(cn=%v)(objectclass=Person))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	person:cn	An LDAP filter that maps the name of a user to an LDAP entry.

EJB Application (ejbApplication)

Defines the properties of an EJB application.

- application-bnd
 - security-role
 - group
 - run-as
 - special-subject

- user
- classloader
 - commonLibrary
 - file
 - fileset
 - folder
 - privateLibrary
 - file
 - fileset
 - folder

Attribute name	Data type	Default value	Description
autoStart	boolean	true	Indicates whether or not the server automatically starts the application.
context-root	string		Context root of an application.
id	string		A unique configuration ID.
location	A file, directory or url.		Location of an application expressed as an absolute path or a path relative to the server-level apps directory.
name	string		Name of an application.
suppressUncoveredHttpMethodWarning	boolean	false	Option to suppress uncovered HTTP method warning message during application deployment.

application-bnd

Binds general deployment information included in the application to specific resources.

false

Attribute name	Data type	Default value	Description
version	string		Version of the application bindings specification.

application-bnd > security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a security role.

application-bnd > security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		Group access ID
id	string		A unique configuration ID.
name	string		Name of a group possessing a security role.

application-bnd > security-role > run-as

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of a user required to access a bean from another bean. The value can be stored in clear text or encoded form. To encode the password, use the securityUtility tool with the encode option.
userid	string		ID of a user required to access a bean from another bean.

application-bnd > security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE. EVERYONE Everyone ALL_AUTHENTICATED_USERS All authenticated users

application-bnd > security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a user possessing a security role.

classloader

Defines the settings for an application classloader.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
classProviderRef	List of references to top level resourceAdapter elements (comma-separated string).		List of class provider references. When searching for classes or resources, this class loader will delegate to the specified class providers after searching its own class path.
commonLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are shared with other classloaders.
delegation	<ul style="list-style-type: none"> • parentFirst • parentLast 	parentFirst	Controls whether parent classloader is used before or after this classloader. If parent first is selected then delegate to immediate parent before searching the classpath. If parent last is selected then search the classpath before delegating to the immediate parent. parentFirst Parent first parentLast Parent last
privateLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

classloader > commonLibrary

List of library references. Library class instances are shared with other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > commonLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > commonLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > commonLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

classloader > privateLibrary

List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > privateLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > privateLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > privateLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

EJB Container (ejbContainer)

Defines the behavior of the EJB container.

- asynchronous
 - contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
- timerService
 - persistentExecutor
 - contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext

Attribute name	Data type	Default value	Description
cacheCleanupInterval	A period of time with second precision	3s	The interval between passivating unused stateful session bean instances when the size is exceeded. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
cacheSize	int Minimum: 1	2053	The number of stateful session bean instances that should be cached in memory.
poolCleanupInterval	A period of time with second precision	30s	The interval between removing unused bean instances. This setting only applies to stateless session and message-driven beans. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
startEJBsAtAppStart	boolean		Specifies when EJB types will be initialized. If this property is set to true, EJB types will be initialized at the time applications are first started. If this property is set to false, EJB types will be initialized at the time the EJB type is first used by an application. If this property is not set, the behavior is determined on a bean-by-bean basis from the start-at-app-start attribute in the ibm-ejb-jar-ext.xml file. This setting does not apply to either message-driven or startup singleton beans. Message-driven and startup singleton beans will always be initialized at the time applications are started.

asynchronous

Defines the behavior of EJB asynchronous methods.

false

Attribute name	Data type	Default value	Description
contextServiceRef	A reference to top level contextService element (string).		The context service used to manage context propagation to asynchronous EJB method threads.
maxUnclaimedRemoteResults	sint Minimum: 1	1000	The maximum number of unclaimed results that the server retains from all remote asynchronous method calls that return a Future object. If the maximum is exceeded, the server purges the result of the method call that completed the longest ago to prevent memory leakage.
unclaimedRemoteResultTimeout	A period of time with second precision	24h	The amount of time that the server retains the result for each remote asynchronous method call that returns a Future object. If an application does not claim the result within the specified period of time, the server purges the result of that method call to prevent memory leakage. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

asynchronous > contextService

The context service used to manage context propagation to asynchronous EJB method threads.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

asynchronous > contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

asynchronous > contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

asynchronous > contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

asynchronous > contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

asynchronous > contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

asynchronous > contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

asynchronous > contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

asynchronous > contextService > classloaderContext

Classloader context propagation configuration.

false

asynchronous > contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

asynchronous > contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

asynchronous > contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

asynchronous > contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCMDN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

timerService

Defines the behavior of the EJB timer service.

false

Attribute name	Data type	Default value	Description
lateTimerThreshold	A period of time with minute precision	5m	Number of minutes after the scheduled expiration of a timer that the start of the timer will be considered late. When a timer does start late, a warning message will be logged indicating that the timer has started later than scheduled. The default threshold is 5 minutes and a value of 0 minutes turns off the warning message feature. Specify a positive integer followed by a unit of time, which can be hours (h) or minutes (m). For example, specify 30 minutes as 30m. You can include multiple values in a single entry. For example, 1h30m is equivalent to 90 minutes.
nonPersistentMaxRetries	int Minimum: -1	-1	When a non-persistent timer expires, the timeout callback method is called. This setting controls how many times the EJB container attempts to retry the timer. If the transaction for this callback method fails or is rolled back, the EJB container must retry the timer at least once. The default value is -1, which means the EJB container retries infinitely until the timer is successful. If the value is set to 0, the EJB container does not retry the timer, however, this results in behavior that is not compliant with the EJB specification.

Attribute name	Data type	Default value	Description
nonPersistentRetryInterval	A period of time with second precision	300s	When a non-persistent timer expires, the timeout callback method is called. If the transaction for this callback method fails or is rolled back, the container must retry the timer. The first retry attempt occurs immediately, and subsequent retry attempts are delayed by the number of seconds specified. If the value is set to 0, then all retries occur immediately. If you do not specify a value, the default interval is 300 seconds. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
persistentExecutorRef	A reference to top level persistentExecutor element (string).		Schedules and runs EJB persistent timer tasks.

timerService > persistentExecutor

Schedules and runs EJB persistent timer tasks.

false

Attribute name	Data type	Default value	Description
contextServiceRef	A reference to top level contextService element (string).	DefaultContextService	Configures how context is captured and propagated to threads.
enableTaskExecution	boolean	true	Determines whether or not this instance may run tasks.

Attribute name	Data type	Default value	Description
initialPollDelay	A period of time with millisecond precision	0	Duration of time to wait before this instance might poll the persistent store for tasks to run. A value of -1 delays polling until it is started programmatically. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
pollInterval	A period of time with millisecond precision	-1	Interval between polling for tasks to run. A value of -1 disables all polling after the initial poll. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
pollSize	int Minimum: 1		The maximum number of task entries to find when polling the persistent store for tasks to run. If unspecified, there is no limit.
retryInterval	A period of time with millisecond precision	1m	The amount of time that must pass between the second and subsequent consecutive retries of a failed task. The first retry occurs immediately, regardless of the value of this attribute. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
retryLimit	short Minimum: -1 Maximum: 10000	10	Limit of consecutive retries for a task that has failed or rolled back, after which the task is considered permanently failed and does not attempt further retries. A value of -1 allows for unlimited retries.
taskStoreRef	A reference to top level databaseStore element (string).	defaultDatabaseStore	Persistent store for scheduled tasks.

timerService > persistentExecutor > contextService

Configures how context is captured and propagated to threads.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

timerService > persistentExecutor > contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

timerService > persistentExecutor > contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

timerService > persistentExecutor > contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

timerService > persistentExecutor > contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

timerService > persistentExecutor > contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

timerService > persistentExecutor > contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

timerService > persistentExecutor > contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

timerService > persistentExecutor > contextService > classloaderContext

Classloader context propagation configuration.

false

timerService > persistentExecutor > contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

timerService > persistentExecutor > contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

timerService > persistentExecutor > contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

timerService > persistentExecutor > contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

Enterprise Application (enterpriseApplication)

Defines the properties of an enterprise application.

- application-bnd
 - security-role
 - group
 - run-as
 - special-subject
 - user
- classloader
 - commonLibrary
 - file
 - fileset
 - folder
 - privateLibrary

- file
- fileset
- folder
- resourceAdapter
 - contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - customize

Attribute name	Data type	Default value	Description
autoStart	boolean	true	Indicates whether or not the server automatically starts the application.
defaultClientModule	string		Default client module of an enterprise application.
id	string		A unique configuration ID.
location	A file, directory or url.		Location of an application expressed as an absolute path or a path relative to the server-level apps directory.
name	string		Name of an application.
suppressUncoveredHttpMethodWarning	boolean	false	Option to suppress uncovered HTTP method warning message during application deployment.

application-bnd

Binds general deployment information included in the application to specific resources.

false

Attribute name	Data type	Default value	Description
version	string		Version of the application bindings specification.

application-bnd > security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a security role.

application-bnd > security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		Group access ID
id	string		A unique configuration ID.
name	string		Name of a group possessing a security role.

application-bnd > security-role > run-as

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of a user required to access a bean from another bean. The value can be stored in clear text or encoded form. To encode the password, use the securityUtility tool with the encode option.
userid	string		ID of a user required to access a bean from another bean.

application-bnd > security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE. EVERYONE Everyone ALL_AUTHENTICATED_USERS All authenticated users

application-bnd > security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user possessing a security role.

classloader

Defines the settings for an application classloader.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
classProviderRef	List of references to top level resourceAdapter elements (comma-separated string).		List of class provider references. When searching for classes or resources, this class loader will delegate to the specified class providers after searching its own class path.
commonLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are shared with other classloaders.
delegation	<ul style="list-style-type: none">parentFirstparentLast	parentFirst	Controls whether parent classloader is used before or after this classloader. If parent first is selected then delegate to immediate parent before searching the classpath. If parent last is selected then search the classpath before delegating to the immediate parent. parentFirst Parent first parentLast Parent last

Attribute name	Data type	Default value	Description
privateLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

classloader > commonLibrary

List of library references. Library class instances are shared with other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > commonLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > commonLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.

Attribute name	Data type	Default value	Description
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > commonLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

classloader > privateLibrary

List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > privateLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > privateLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > privateLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

resourceAdapter

Specifies configuration for a resource adapter that is embedded in an application.

false

Attribute name	Data type	Default value	Description
alias	string	\${id}	Overrides the default identifier for the resource adapter. The identifier is used in the name of the resource adapter's configuration properties element, which in turn is used in determining the name of configuration properties elements for any resources provided by the resource adapter. The resource adapter's configuration properties element name has the format, properties.<APP_NAME>.<ALIAS>, where <APP_NAME> is the name of the application and <ALIAS> is the configured alias. If unspecified, the alias defaults to the module name of the resource adapter.
autoStart	boolean		Configures whether a resource adapter starts automatically upon deployment of the resource adapter or lazily upon injection or lookup of a resource.
contextServiceRef	A reference to top level contextService element (string).		Configures how context is captured and propagated to threads.
id	string		Identifies the name of the embedded resource adapter module to which this configuration applies.

resourceAdapter > contextService

Configures how context is captured and propagated to threads.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

resourceAdapter > contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

resourceAdapter > contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

resourceAdapter > contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

resourceAdapter > contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

resourceAdapter > contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

resourceAdapter > contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

resourceAdapter > contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

resourceAdapter > contextService > classloaderContext

Classloader context propagation configuration.

false

resourceAdapter > contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

resourceAdapter > contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

resourceAdapter > contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

resourceAdapter > contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

resourceAdapter > customize

Customizes the configuration properties element for the activation specification, administered object, or connection factory with the specified interface and/or implementation class.

false

Attribute name	Data type	Default value	Description
implementation	string		Fully qualified implementation class name for which the configuration properties element should be customized.
interface	string		Fully qualified interface class name for which the configuration properties element should be customized.

Attribute name	Data type	Default value	Description
suffix	string		Overrides the default suffix for the configuration properties element. For example, "CustomConnectionFactory" in properties.rarModule1.CustomConnectionFactory. The suffix is useful to disambiguate when multiple types of connection factories, administered objects, or endpoint activations are provided by a resource adapter. If a configuration properties element customization omits the suffix or leaves it blank, no suffix is used.

Event Logging (eventLogging)

Logs a record of events, such as the JDBC requests and servlet requests, and their durations.

Attribute name	Data type	Default value	Description
eventTypes	string	all	A list of comma-separated event types that needs to be logged. Use all, to log all event types.
includeContextInfo	boolean	true	Indicates if the context information details are included in the log output.
logMode	<ul style="list-style-type: none"> • entry • entryExit • exit 	exit	<p>Controls whether the event logging occurs at the entry to events, at the exit from events, or both.</p> <p>entry log at entry</p> <p>entryExit log at entry and exit</p> <p>exit log at exit</p>
minDuration	A period of time with millisecond precision	1s	Exit entries will be logged for events longer than minDuration. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
sampleRate	int Minimum: 1	1	To sample one out of every n requests, set sampleRate to n. To sample all requests, set sampleRate to 1.

Executor Management (executor)

Defines settings for the Liberty kernel default executor. Note that there is always one and exactly one default executor, for use by the Liberty runtime only and not directly accessible by applications. Applications that need to configure and utilize executors should instead use Managed Executors.

Attribute name	Data type	Default value	Description
coreThreads	int	-1	Steady-state or core number of threads to associate with the executor. The number of threads associated with the executor will quickly grow to this number. If this value is less than 0, a default value is used. This default value is calculated based on the number of hardware threads on the system.
keepAlive	A period of time with millisecond precision	60s	Amount of time to keep an idle thread in the pool before allowing it to terminate. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
maxThreads	int	-1	Maximum number of threads that can be associated with the executor. If greater than 0, this value must be greater than or equal to the value of coreThreads. If the value of maxThreads is less than or equal to 0, the maximum number of threads is unbounded. Note that the actual number of threads associated with the executor is determined dynamically by the Liberty kernel, so leaving the maximum threads unbounded does not imply that the runtime will actively create large amounts of threads; it simply lets the Liberty kernel decide how many threads to associate with the executor without having a defined upper boundary.
name	string	Default Executor	The name of the Liberty kernel default executor.
rejectedWorkPolicy	<ul style="list-style-type: none"> • CALLER_RUNS • ABORT 	ABORT	<p>Policy to employ when the executor is unable to stage work for execution.</p> <p>CALLER_RUNS Execute the work immediately on the caller's thread.</p> <p>ABORT Raise an exception.</p>

Attribute name	Data type	Default value	Description
stealPolicy	<ul style="list-style-type: none"> • STRICT • NEVER • LOCAL 	LOCAL	<p>The work-stealing policy to employ. The options for this policy determine how work is queued, and how threads obtain queued work.</p> <p>STRICT All threads that generate work own a local work pile. Threads that are associated with the executor take work from other threads when the local work pile is exhausted.</p> <p>NEVER A global work queue is used to feed work to threads that are associated with the executor. No stealing will occur.</p> <p>LOCAL A global work queue is used for work that is generated by threads that are not associated with the executor. Work generated by threads associated with the executor is placed on a local work pile. This work pile is owned by the generating thread, unless another thread steals it. Threads that are associated with the executor take work associated with other threads if the local work pile is empty and there is no work on the global work queue.</p>

Feature Manager (featureManager)

Defines how the server loads features.

- feature

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Action to take after a failure to load a feature.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

feature

Specifies a feature to be used when the server runs.

false

string

User Registry Federation (federatedRepository)

Configuration for the user registry federation.

- extendedProperty
- primaryRealm
 - defaultParents
 - groupDisplayNameMapping
 - groupSecurityNameMapping
 - participatingBaseEntry
 - uniqueGroupIdMapping
 - uniqueUserIdMapping
 - userDisplayNameMapping
 - userSecurityNameMapping
- realm
 - defaultParents
 - groupDisplayNameMapping
 - groupSecurityNameMapping
 - participatingBaseEntry

- uniqueGroupIdMapping
- uniqueUserIdMapping
- userDisplayNameMapping
- userSecurityNameMapping
- supportedEntityType
 - defaultParent
 - name

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
maxSearchResults	int	4500	Maximum number of entries that can be returned in a search.
pageCacheSize	int	1000	Defines the number of pagination requests that can be stored in the cache. The paging cache size needs to be configured based on the number of pagination requests executed on the system and the hardware system resources available.
pageCacheTimeout	A period of time with millisecond precision	30000ms	Defines the maximum time that an entry, which added to the page cache, is available. When the specified time has elapsed, the entry from the page cache is cleared. This needs to be configured based on the interval between pagination search requests executed on the system and the hardware system resources available. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Attribute name	Data type	Default value	Description
searchTimeout	A period of time with millisecond precision	10m	Maximum amount of time, in milliseconds, to process a search. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

extendedProperty

The extended properties for Person and Group.

false

Attribute name	Data type	Default value	Description
dataType	<ul style="list-style-type: none"> • Double • Long • Date • String • BigDecimal • BigInteger • Boolean • Integer 		<p>Defines the data type of the property extended for Person and Group. The basic Java data types are supported.</p> <p>Double Double</p> <p>Long Long</p> <p>Date Date</p> <p>String String</p> <p>BigDecimal BigDecimal</p> <p>BigInteger BigInteger</p> <p>Boolean Boolean</p> <p>Integer Integer</p>
defaultValue	string		Defines the default value for the property during write operation, if no default value is set.
entityType	<ul style="list-style-type: none"> • PersonAccount • Group 		<p>The name of the entity being mapped. The name of the entity can be PersonAccount or Group.</p> <p>PersonAccount Person</p> <p>Group Group</p>
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
multiValued	boolean	false	Defines if the property extended for Person and Group supports multiple values.
name	string		Defines the name of the property extended for Person and Group.

primaryRealm

Primary realm configuration.

false

Attribute name	Data type	Default value	Description
allowOpIfRepoDown	boolean	false	Specifies whether operation is allowed if a repository is down. The default value is false.
delimiter	string	/	Delimiter used to qualify the realm under which the operation should be executed. For example, userid=test1/myrealm where / is the delimiter and myrealm is the realm name.
name	string		Name of the realm.

primaryRealm > defaultParents

The default parent mapping for the realm.

false

Attribute name	Data type	Default value	Description
name	string		The name of the entity being mapped. The name of the entity can be PersonAccount or Group.
parentUniqueName	string		The distinguished name under Base distinguished name (DN) in the repository under which all entities of the configured type will be created.

primaryRealm > groupDisplayNameMapping

The input and output property mappings for group display name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	cn	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

primaryRealm > groupSecurityNameMapping

The input and output property mappings for group security name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	cn	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

primaryRealm > participatingBaseEntry

The Base Entry that is part of this realm.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of the base entry.

primaryRealm > uniqueGroupIdMapping

The input and output property mappings for unique group id in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

primaryRealm > uniqueUserIdMapping

The input and output property mappings for unique user id used in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	uniqueName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

primaryRealm > userDisplayNameMapping

The input and output property mappings for user display name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	principalName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

Attribute name	Data type	Default value	Description
outputProperty	string	principalName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

primaryRealm > userSecurityNameMapping

The input and output property mappings for user security name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	principalName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm

Reference to the realm.

false

Attribute name	Data type	Default value	Description
allowOpIfRepoDown	boolean	false	Specifies whether operation is allowed if a repository is down. The default value is false.
delimiter	string	/	Delimiter used to qualify the realm under which the operation should be executed. For example, userid=test1/myrealm where / is the delimiter and myrealm is the realm name.
id	string		A unique configuration ID.
name	string		Name of the realm.

realm > defaultParents

The default parent mapping for the realm.

false

Attribute name	Data type	Default value	Description
name	string		The name of the entity being mapped. The name of the entity can be PersonAccount or Group.
parentUniqueName	string		The distinguished name under Base distinguished name (DN) in the repository under which all entities of the configured type will be created.

realm > groupDisplayNameMapping

The input and output property mappings for group display name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	cn	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm > groupSecurityNameMapping

The input and output property mappings for group security name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

Attribute name	Data type	Default value	Description
outputProperty	string	cn	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm > participatingBaseEntry

The Base Entry that is part of this realm.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of the base entry.

realm > uniqueGroupIdMapping

The input and output property mappings for unique group id in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	cn	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm > uniqueUserIdMapping

The input and output property mappings for unique user id used in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	uniqueName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

Attribute name	Data type	Default value	Description
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm > userDisplayNameMapping

The input and output property mappings for user display name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	principalName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	principalName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

realm > userSecurityNameMapping

The input and output property mappings for user security name in an user registry operation.

false

Attribute name	Data type	Default value	Description
inputProperty	string	principalName	The property that maps to the user registry attribute for input. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.
outputProperty	string	uniqueName	The property that maps to the user registry attribute for output. The valid values are: uniqueId, uniqueName, externalId, externalName and the attributes of PersonAccount and Group entity types.

supportedEntityType

The default parent for an entity type mapping.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

supportedEntityType > defaultParent

The distinguished name under Base distinguished name (DN) in the repository under which all entities of the configured type will be created.

false

string

supportedEntityType > name

The name of the entity being mapped. The name of the entity can be PersonAccount or Group.

false

string

Fileset (fileset)

Specifies a set of files starting from a base directory and matching a set of patterns.

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Host Authentication Information (hostAuthInfo)

Connection details to allow for the collective controller to authenticate to the server's host.

Attribute name	Data type	Default value	Description
rpcHost	string	localhost	The fully qualified host name or IP address. A '*' wildcard will result in host name detection; this is not recommended for multi-homed systems and may result in unexpected behaviour. The host name must be unique within the network and must be the host name on which the remote connection protocol is listening (SSH, or OS specific RPC). This value will inherit from the defaultHostName variable if not set. The host name set here will directly control where the server's information is stored within the collective controller repository.
rpcPort	int	22	The port on which the remote connection protocol is listening (SSH, or OS specific RPC). See product documentation for supported RPC mechanisms.
rpcUser	string		The operating system user ID to use to connect to the host.

Attribute name	Data type	Default value	Description
rpcUserHome	string		The home directory of the user login ID. Only required to be set if sudo is to be used and SSH generation is to be done automatically.
rpcUserPassword	Reversably encoded password (string)		The password for the operating system user. If this property is not set, key-based authentication will be used. Use of key-based authentication is recommended for hosts which support SSH. If this property is set and sshPrivateKeyPath is also set, the key will take precedence.
sshPrivateKeyPassword	Reversably encoded password (string)		The password for the SSH private key.
sshPrivateKeyPath	string		The path to the SSH private key file. If the key pair does not exist, a key pair will be generated automatically. The private key is required for key-based authentication.
sshPublicKeyPath	string		The path to the SSH public key file. If the key pair does not exist, a key pair will be generated automatically. The public key will be placed into the configured userId's authorized_keys file if it is not present. Setting the path to the public key is not required.
sudoUser	string		The sudo user ID. This property should not be set when useSudo=false.
sudoUserPassword	Reversably encoded password (string)		The password for the sudo user. This property should not be set when useSudo=false.
useHostCredentials	boolean		If this property is set to true, then the product uses the RPC credentials of the host to invoke commands and ignores other parameters in the 'Host Authentication Information' element.

Attribute name	Data type	Default value	Description
useSudo	boolean		If this property is set to true, then sudo will be used to invoke commands. The user to sudo as can be controlled by setting sudoUser attribute. If sudoUser is not set, then the user to sudo as will be the configured default sudo user for the host. If this property is not set, and either sudoUser or sudoUserPassword are set, then useSudo is assumed to be true. If this property is set to false, and either sudoUser or sudoUserPassword are set, then a warning will be printed and the sudo options will be ignored.

Host Singleton (hostSingleton)

Host singleton elector configuration

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string	*	The name of the singleton. A '*' wildcard is the default value and indicates this configuration applies to all singletons in this server.
port	int	0	The port to use for host singleton leader elections. A value of 0 is the default and means that no election will occur. In this case, the singleton in each member will be its own leader.

HTTP Access Logging (httpAccessLogging)

HTTP access logs contain a record of all inbound HTTP client requests.

Attribute name	Data type	Default value	Description
enabled	boolean	true	Enable access logging.
filePath	Path to a file	\${server.output.dir}/logs/http_access.log	Directory path and name of the access log file. Standard variable substitutions, such as \${server.output.dir}, can be used when specifying the directory path.
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
logFormat	string	%h %u %{t}W "%r" %s %b	Specifies the log format that is used when logging client access information.
maxFileSize	int Minimum: 0	20	Maximum size of a log file, in megabytes, before being rolled over; a value of 0 means no limit.
maxFiles	int Minimum: 0	2	Maximum number of log files that will be kept, before the oldest file is removed; a value of 0 means no limit.

HTTP Dispatcher (httpDispatcher)

HTTP Dispatcher configuration.

- trustedHeaderOrigin

Attribute name	Data type	Default value	Description
appOrContextRootMissingMessage	string		Message to return to the client when the application in the requested URI can not be found.
enableWelcomePage	boolean	true	Enables the default Liberty profile welcome page when no application is bound to a context root of "/". The default value is true.

trustedHeaderOrigin

Private headers are used by the web server plug-in to provide information about the original request. These headers take precedence over the http Host header, and are used to select a virtual host to service a request. The default value is '*', which will trust incoming private headers from any source. Specify 'none' to disable private headers and rely only on the http Host header, or specify a list of IP addresses to restrict private header processing to specific trusted sources.

false

string

HTTP Transport Encoding (httpEncoding)

HTTP transport encoding settings

Attribute name	Data type	Default value	Description
converter.Big5	string	Cp950	Big5 Chinese converter
converter.EUC-JP	string	Cp33722C	EUC Japanese converter (EUC-JP)
converter.EUC-KR	string	Cp970	EUC Korean converter (EUC-KR)
converter.EUC-TW	string	Cp964	EUC Chinese (Taiwan) converter (EUC-TW)
converter.EUC_KR	string	Cp970	EUC Korean converter (EUC_KR)

Attribute name	Data type	Default value	Description
converter.GB2312	string	EUC_CN	GB2312 Chinese converter
converter.ISO-2022-KR	string	ISO2022KR	ISO-2022 Korean converter (ISO-2022-KR)
converter.Shift_JIS	string	Cp943C	Shift_JIS Japanese converter
encoding.ar	string	ISO-8859-6	Arabic language encoding (ar)
encoding.be	string	ISO-8859-5	Belarusian language encoding (be)
encoding.bg	string	ISO-8859-5	Bulgarian language encoding (bg)
encoding.bn	string	UTF-8	Bengali language encoding (bn)
encoding.ca	string	ISO-8859-1	Catalan language encoding (ca)
encoding.cs	string	ISO-8859-2	Czech language encoding (cs)
encoding.da	string	ISO-8859-1	Danish language encoding (da)
encoding.de	string	ISO-8859-1	German language encoding (de)
encoding.el	string	ISO-8859-7	Greek language encoding (el)
encoding.en	string	ISO-8859-1	English language encoding (en)
encoding.es	string	ISO-8859-1	Spanish language encoding (es)
encoding.et	string	ISO-8859-4	Estonian language encoding (et)
encoding.eu	string	ISO-8859-1	Basque language encoding (eu)
encoding.fa	string	ISO-8859-6	Persian language encoding (fa)
encoding.fi	string	ISO-8859-1	Finnish language encoding (fi)
encoding.fo	string	ISO-8859-2	Faroese language encoding (fo)
encoding.fr	string	ISO-8859-1	French language encoding (fr)
encoding.he	string	ISO-8859-8	Hebrew language encoding (he)
encoding.hi	string	UTF-8	Hindi language encoding (hi)
encoding.hr	string	ISO-8859-2	Croatian language encoding (hr)
encoding.hu	string	ISO-8859-2	Hungarian language encoding (hu)
encoding.hy	string	UTF-8	Armenian language encoding (hy)

Attribute name	Data type	Default value	Description
encoding.is	string	ISO-8859-1	Icelandic language encoding (is)
encoding.it	string	ISO-8859-1	Italian language encoding (it)
encoding.iw	string	ISO-8859-8	Hebrew language encoding (iw)
encoding.ja	string	Shift_JIS	Japanese language encoding (ja)
encoding.ji	string	ISO-8859-8	Yiddish language encoding (ji)
encoding.ka	string	UTF-8	Georgian language encoding (ka)
encoding.ko	string	EUC-KR	Korean language encoding (ko)
encoding.lt	string	ISO-8859-2	Lithuanian language encoding (lt)
encoding.lv	string	ISO-8859-4	Latvian language encoding (lv)
encoding.mk	string	ISO-8859-5	Macedonian language encoding (mk)
encoding.mr	string	UTF-8	Marathi language encoding (mr)
encoding.ms	string	ISO-8859-6	Malay language encoding (ms)
encoding.mt	string	ISO-8859-3	Maltese language encoding (mt)
encoding.nl	string	ISO-8859-1	Dutch language encoding (nl)
encoding.no	string	ISO-8859-1	Norwegian language encoding (no)
encoding.pl	string	ISO-8859-2	Polish language encoding (pl)
encoding.pt	string	ISO-8859-1	Portuguese language encoding (pt)
encoding.ro	string	ISO-8859-2	Romanian language encoding (ro)
encoding.ru	string	ISO-8859-5	Russian language encoding (ru)
encoding.sa	string	UTF-8	Sanskrit language encoding (sa)
encoding.sh	string	ISO-8859-2	Serbo-Croatian language encoding (sh)
encoding.sk	string	ISO-8859-2	Slovak language encoding (sk)
encoding.sl	string	ISO-8859-2	Slovenian language encoding (sl)
encoding.sq	string	ISO-8859-2	Albanian language encoding (sq)

Attribute name	Data type	Default value	Description
encoding.sr	string	ISO-8859-5	Serbian language encoding (sr)
encoding.sv	string	ISO-8859-1	Swedish language encoding (sv)
encoding.ta	string	UTF-8	Tamil language encoding (ta)
encoding.th	string	windows-874	Thai language encoding (th)
encoding.tr	string	ISO-8859-9	Turkish language encoding (tr)
encoding.uk	string	ISO-8859-5	Ukrainian language encoding (uk)
encoding.vi	string	windows-1258	Vietnamese language encoding (vi)
encoding.yi	string	ISO-8859-8	Yiddish language encoding (yi)
encoding.zh	string	GB2312	Chinese language encoding (zh)
encoding.zh_TW	string	Big5	Chinese language encoding (zh_TW)

HTTP Endpoint (httpEndpoint)

Configuration properties for an HTTP endpoint.

- accessLogging
- httpOptions
- sslOptions
- tcpOptions

Attribute name	Data type	Default value	Description
accessLoggingRef	A reference to top level httpAccessLogging element (string).		HTTP access logging configuration for the endpoint.
enabled	boolean	true	Toggle the availability of an endpoint. When true, this endpoint will be activated by the dispatcher to handle HTTP requests.
host	string	localhost	IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource. Use '*' for all available network interfaces.
httpOptionsRef	A reference to top level httpOptions element (string).	defaultHttpOptions	HTTP protocol options for the endpoint.

Attribute name	Data type	Default value	Description
httpPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests. Use -1 to disable this port.
httpsPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests secured with SSL (https). Use -1 to disable this port.
id	string		A unique configuration ID.
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Action to take after a failure to start an endpoint.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>
sslOptionsRef	A reference to top level sslOptions element (string).		SSL protocol options for the endpoint.
tcpOptionsRef	A reference to top level tcpOptions element (string).	defaultTCPOptions	TCP protocol options for the endpoint.

accessLogging

HTTP access logging configuration for the endpoint.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	Enable access logging.
filePath	Path to a file	\${server.output.dir}/logs/http_access.log	Directory path and name of the access log file. Standard variable substitutions, such as \${server.output.dir}, can be used when specifying the directory path.
logFormat	string	%h %u %{t}W "%r" %s %b	Specifies the log format that is used when logging client access information.

Attribute name	Data type	Default value	Description
maxFileSize	int Minimum: 0	20	Maximum size of a log file, in megabytes, before being rolled over; a value of 0 means no limit.
maxFiles	int Minimum: 0	2	Maximum number of log files that will be kept, before the oldest file is removed; a value of 0 means no limit.

httpOptions

HTTP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
keepAliveEnabled	boolean	true	Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.
maxKeepAliveRequests	int Minimum: -1	100	Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.
persistTimeout	A period of time with second precision	30s	Amount of time that a socket will be allowed to remain idle between requests. This setting only applies if persistent connections are enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
readTimeout	A period of time with second precision	60s	Amount of time to wait for a read request to complete on a socket after the first read occurs. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
removeServerHeader	boolean	false	Removes server implementation information from HTTP headers and also disables the default Liberty profile welcome page.
writeTimeout	A period of time with second precision	60s	Amount of time to wait on a socket for each portion of the response data to be transmitted. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

sslOptions

SSL protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
sessionTimeout	A period of time with second precision	1d	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
sslRef	A reference to top level ssl element (string).		The default SSL configuration repertoire. The default value is defaultSSLSettings.
suppressHandshakeErrors	boolean	false	Disable logging of SSL handshake errors. SSL handshake errors can occur during normal operation, however these messages can be useful when SSL is behaving unexpectedly.

tcpOptions

TCP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
inactivityTimeout	A period of time with millisecond precision	60s	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
soReuseAddr	boolean	true	Enables immediate rebind to a port with no active listener.

HTTP Options (httpOptions)

HTTP protocol configuration.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
keepAliveEnabled	boolean	true	Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.

Attribute name	Data type	Default value	Description
maxKeepAliveRequests	int Minimum: -1	100	Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.
persistTimeout	A period of time with second precision	30s	Amount of time that a socket will be allowed to remain idle between requests. This setting only applies if persistent connections are enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
readTimeout	A period of time with second precision	60s	Amount of time to wait for a read request to complete on a socket after the first read occurs. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
removeServerHeader	boolean	false	Removes server implementation information from HTTP headers and also disables the default Liberty profile welcome page.
writeTimeout	A period of time with second precision	60s	Amount of time to wait on a socket for each portion of the response data to be transmitted. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

HTTP Proxy Redirect (httpProxyRedirect)

Configures port redirection. HTTP Proxy Redirect is used when redirecting HTTP requests from a non-secure port (for example, 80) to an SSL-enabled secured port (for example, 443).

Attribute name	Data type	Default value	Description
enabled	boolean	true	This attribute determines whether or not the server should redirect ports that are specified in this configuration element. The default is true.
host	string	*	The host name used for this proxy redirect. The server redirects HTTP requests only if the incoming request specifies a host name that matches this value. The default is * (all hosts).
httpPort	int Minimum: 1 Maximum: 65535		The (non-secure) port to redirect from. Incoming HTTP requests on this port are redirected to the specified HTTPS port.
httpsPort	int Minimum: 1 Maximum: 65535		The (secure) port to redirect to. Incoming HTTP requests that use the HTTP port are redirected to this port.
id	string		A unique configuration ID.

HTTP Session (httpSession)

Configuration for HTTP session management.

Attribute name	Data type	Default value	Description
allowOverflow	boolean	true	Allows the number of sessions in memory to exceed the value of the Max in-memory session count property.
alwaysEncodeUrl	boolean	false	The Servlet 2.5 specification specifies to not encode the URL on a response.encodeURL call if it is not necessary. To support backward compatibility when URL encoding is enabled, set this property to true to call the encodeURL method. The URL is always encoded, even if the browser supports cookies.

Attribute name	Data type	Default value	Description
cloneId	string		The clone identifier of the cluster member. Within a cluster, this identifier must be unique to maintain session affinity. When set, this name overwrites the default name generated by the server.
cloneSeparator	string	:	The single character used to separate the session identifier from the clone identifier in session cookies. The default value should usually be used. On some Wireless Application Protocol (WAP) devices, a colon (:) is not allowed, so a plus sign (+) should be used instead. Different values should rarely be used. You should understand the clone character requirements of other products running on your system before using this property to change the clone separator character. The fact that any character can be specified as the value for this property does not imply that the character you specify will function correctly. This fact also does not imply that IBM is responsible for fixing any problem that might arise from using an alternative character.
cookieDomain	string		Domain field of a session tracking cookie.
cookieHttpOnly	boolean	true	Specifies that session cookies include the HttpOnly field. Browsers that support the HttpOnly field do not enable cookies to be accessed by client-side scripts. Using the HttpOnly field will help prevent cross-site scripting attacks.

Attribute name	Data type	Default value	Description
cookieMaxAge	A period of time with second precision	-1	Maximum amount of time that a cookie can reside on the client browser. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
cookieName	string	JSESSIONID	A unique name for a session management cookie.
cookiePath	string	/	A cookie is sent to the URL designated in the path.
cookieSecure	boolean	false	Specifies that the session cookies include the secure field.
cookiesEnabled	boolean	true	Specifies that session tracking uses cookies to carry session identifiers.
debugCrossover	boolean	false	Enable this option to perform additional checks to verify that only the session associated with the request is accessed or referenced, and log messages if any discrepancies are detected. Disable this option to skip the additional checks.

Attribute name	Data type	Default value	Description
forceInvalidationMultiple	int	3	If your requests normally are not bound by a response time limit, specify 0 to indicate that the session manager should wait indefinitely until a request is complete before attempting to invalidate the session. Otherwise, set this property to a positive integer to delay the invalidation of active sessions. Active timed out sessions will not be invalidated by the first invalidation interval pass, but will be invalidated by the interval pass based on this value. For example, a value of 2 would invalidate an active session on the second invalidation interval pass after the session timeout has expired.
idLength	int	23	Length of the session identifier.
idReuse	boolean	false	In a multi-JVM environment that is not configured for session persistence, setting this property to "true" enables the session manager to use the same session information for all of a user's requests even if the web applications that are handling these requests are governed by different JVMs. The default value for this property is false. Set this property to true if you want to enable the session manager to use the session identifier sent from a browser to preserve session data across web applications that are running in an environment that is not configured for session persistence.

Attribute name	Data type	Default value	Description
invalidateOnUnauthorizedSessionRequestException	boolean	false	Set this property to true if, in response to an unauthorized request, you want the session manager to invalidate a session instead of issuing an UnauthorizedSessionRequestException. When a session is invalidated, the requester can create a new session, but does not have access to any of the previously saved session data. This allows a single user to continue processing requests to other applications after a logout while still protecting session data.
invalidationTimeout	A period of time with second precision	30m	Amount of time a session can go unused before it is no longer valid. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxInMemorySessionCount	int	1000	Maximum number of sessions to maintain in memory for each web module.
noAdditionalInfo	boolean	false	Forces removal of information that is not needed in session identifiers.
protocolSwitchRewritingEnabled	boolean	false	Adds the session identifier to a URL when the URL requires a switch from HTTP to HTTPS or from HTTPS to HTTP.

Attribute name	Data type	Default value	Description
reaperPollInterval	A period of time with second precision	-1	The wake-up interval, in seconds, for the process that removes invalid sessions. The minimum value is 30 seconds. If a value less than the minimum is entered, an appropriate value is automatically determined and used. This value overrides the default installation value, which is between 30 and 360 seconds, based off the session timeout value. Because the default session timeout is 30 minutes, the reaper interval is usually between 2 and 3 minutes. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
rewriteId	string	jsessionId	Use this property to change the key used with URL rewriting.
securityIntegrationEnabled	boolean	true	Enables security integration, which causes the session management facility to associate the identity of users with their HTTP sessions.
securityUserIgnoreCase	boolean	false	Indicates that the session security identity and the client security identity should be considered a match even if their cases are different. For example, when this property is set to true, the session security identity USER1 matches the client security identities User1 and user1.
sslTrackingEnabled	boolean	false	Specifies that session tracking uses Secure Sockets Layer (SSL) information as a session identifier.

Attribute name	Data type	Default value	Description
urlRewritingEnabled	boolean	false	Specifies that the session management facility uses rewritten URLs to carry the session identifiers.
useContextRootAsCookiePath	boolean	false	Specifies that the cookie path equals the context root of the web module instead of /

HTTP Session Database (httpSessionDatabase)

Controls how HTTP sessions are persisted to a database.

Attribute name	Data type	Default value	Description
dataSourceRef	string		The identifier of the data source the session manager should use to persist HTTP session data.

Attribute name	Data type	Default value	Description
db2RowSize	<ul style="list-style-type: none"> • 32KB • 4KB • 8KB • 16KB 	4KB	<p>Table space page size configured for the sessions table, if using a DB2 database. Increasing this value can improve database performance in some environments.</p> <p>32KB Use a table space page size of 32 KB. You must additionally create a DB2 buffer pool and table space, and specify 32KB as the page size for both. You must also specify the name of the table space you created.</p> <p>4KB Use the default table space page size of 4 KB. You do not need to create a DB2 buffer pool or table space, and you do not need to specify a table space name.</p> <p>8KB Use a table space page size of 8 KB. You must additionally create a DB2 buffer pool and table space, and specify 8KB as the page size for both. You must also specify the name of the table space you created.</p> <p>16KB Use a table space page size of 16 KB. You must additionally create a DB2 buffer pool and table space, and specify 16KB as the page size for both. You must also specify the name of the table space you created.</p>

Attribute name	Data type	Default value	Description
noAffinitySwitchBack	boolean	false	Set this property to "true" to maintain affinity to the new member even after original one comes back up. When a cluster member fails, its requests routed to a different cluster member, and sessions are activated in that other member. Thus, session affinity is maintained to the new member, and when failed cluster member comes back up, the requests for sessions that were created in the original cluster member are routed back to it. Allowed values are true or false, with the default being false. Set this property to true when you have distributed sessions configured with time-based write. Note that this property has no affect on the behavior when distributed sessions are not enabled.
onlyCheckInCacheDuringPrehook	boolean	false	A value of true indicates that the last access time of a session should only be updated if a request gets the session. A value of false indicates that the last access time of a session should be updated upon every request. Changing this value can improve performance in some environments.

Attribute name	Data type	Default value	Description
optimizeCacheIdIncrements	boolean	true	If the user's browser session is moving back and forth across multiple web applications, you might see extra persistent store activity as the in-memory sessions for a web module are refreshed from the persistent store. As a result, the cache identifiers are continually increasing and the in-memory session attributes are overwritten by those of the persistent copy. Set this property to true if you want to prevent the cache identifiers from continually increasing. A value of true indicates that the session manager should assess whether the in-memory session for a web module is older than the copy in persistent store. If the configuration is a cluster, ensure that the system times of each cluster member are as identical as possible.
scheduleInvalidation	boolean	false	Enable this option to reduce the number of database updates required to keep the HTTP sessions alive. Specify the two hours of a day when there is the least activity in the application server. When this option is disabled, the invalidator process runs every few minutes to remove invalidated HTTP sessions.
scheduleInvalidationFirstHour	int	0	Indicates the first hour during which the invalidated sessions are cleared from the persistent store. Specify this value as an integer between 0 and 23. This value is valid only when schedule invalidation is enabled.

Attribute name	Data type	Default value	Description
scheduleInvalidationSecondHour	integer	0	Indicates the second hour during which the invalidated sessions are cleared from the persistent store. Specify this value as an integer between 0 and 23. This value is valid only when schedule invalidation is enabled.
skipIndexCreation	boolean	false	Set this property to "true" to disable index creation on server startup. This custom property should only be used if you want to manually create your own database indices for session persistence. However, it is recommended that you let session manager create database indices. Before enabling this property, make sure that the correct index does exist on your session database.
tableName	string	sessions	The database table name.
tableSpaceName	string		Table space to be used for the sessions table. This value is only required when the DB2 Row Size is greater than 4KB.
useInvalidatedId	boolean	true	Set this property to "true" to reuse the incoming identifier if the session with that identifier was recently invalidated. This is a performance optimization because it prevents checking the persistent store.
useMultiRowSchema	boolean	false	When enabled, each session data attribute is placed in a separate row in the database, allowing larger amounts of data to be stored for each session. This configuration can yield better performance when session attributes are very large and few changes are required to the session attributes. When disabled, all session data attributes are placed in the same row for each session.

Attribute name	Data type	Default value	Description
useOracleBlob	boolean	false	Set this property to "true" to create the database table using the Binary Large Object (BLOB) data type for the medium column. This value increases performance of persistent sessions when Oracle databases are used. Due to an Oracle restriction, BLOB support requires use of the Oracle Call Interface (OCI) database driver for more than 4000 bytes of data. You must also ensure that a new sessions table is created before the server is restarted by dropping your old sessions table or by changing the datasource definition to reference a database that does not contain a sessions table.
usingCustomSchemaName	boolean	false	Set this property to "true" if you are using DB2 for session persistence and the currentSchema property is set in the data source.
writeContents	<ul style="list-style-type: none"> • ALL_SESSION_ATTRIBUTES • ONLY_UPDATED_ATTRIBUTES 	ONLY_UPDATED_ATTRIBUTES	<p>Specifies how much session data should be written to the persistent store. By default, only updated attributes are written, but all attributes can be written instead (regardless of whether or not they changed).</p> <p>ALL_SESSION_ATTRIBUTES All attributes are written to the persistent store.</p> <p>ONLY_UPDATED_ATTRIBUTES Only updated attributes are written to the persistent store.</p>

Attribute name	Data type	Default value	Description
writeFrequency	<ul style="list-style-type: none"> • TIME_BASED_WRITE • END_OF_SERVLET_SERVICE • MANUAL_UPDATE 	END_OF_SERVLET_SERVICE	<p>Specifies when session data is written to the persistent store. By default, session data is written to the persistent store after the servlet completes execution. Changing this value can improve performance in some environments.</p> <p>TIME_BASED_WRITE Session data is written to the persistent store based on the specified write interval value.</p> <p>END_OF_SERVLET_SERVICE Session data is written to the persistent store after the servlet completes execution.</p> <p>MANUAL_UPDATE A programmatic sync on the IBMSession object is required to write the session data to the persistent store.</p>
writeInterval	A period of time with second precision	2m	Number of seconds that should pass before writing session data to the persistent store. The default is 120 seconds. This value is only used when a time based write frequency is enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

HTTP Whiteboard (httpWhiteboard)

The HTTP Whiteboard provides a runtime environment for hosting servlets and resources provided by OSGi services.

Attribute name	Data type	Default value	Description
contextPath	string	/osgi/http	The context path to use for the HTTP Whiteboard runtime environment.

IBM Tivoli Directory Server LDAP Filters (idsLdapFilterProperties)

Specifies the list of default IBM Tivoli Directory Server LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)(objectclass=groupOfUniqueNames)))(objectclass=groupOfNames)	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	An LDAP filter that maps the name of a group to its members.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

Include (include)

Specify a configuration resource to include in the server's configuration.

Attribute name	Data type	Default value	Description
location	A file, directory or url.		Specifies the resource location. This can be a file path or a URI for a remote resource.

Attribute name	Data type	Default value	Description
onConflict	<ul style="list-style-type: none"> • IGNORE • REPLACE • MERGE 	MERGE	<p>Specifies the behavior that is used to merge elements when conflicts are found.</p> <p>IGNORE Conflicting elements in the included file will be ignored.</p> <p>REPLACE When elements conflict, the element from the included file will replace the conflicting element.</p> <p>MERGE Conflicting elements will be merged together.</p>
optional	boolean	false	Allow the included resource to be skipped if it cannot be found.

Sun Java System Directory Server LDAP Filters (iplanetLdapFilterProperties)

Specifies the list of default Sun Java System Directory Server LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=ldapsubentry))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	nsRole:nsRole	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=inetOrgPerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	inetOrgPerson:uid	An LDAP filter that maps the name of a user to an LDAP entry.

JAAS Login Context Entry (jaasLoginContextEntry)

The JAAS login context entry configuration.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
loginModuleRef	List of references to top level jaasLoginModule elements (comma-separated string).	hashtable,userNameAndPassword,certificate,lockID	Word, certificate, lock ID of a JAAS login module.
name	string		Name of a JAAS configuration entry.

JAAS Login Module (jaasLoginModule)

A login module in the JAAS configuration.

- library
 - file
 - fileset
 - folder
- options

Attribute name	Data type	Default value	Description
className	string		Fully-qualified package name of the JAAS login module class.

Attribute name	Data type	Default value	Description
controlFlag	<ul style="list-style-type: none"> • SUFFICIENT • REQUISITE • REQUIRED • OPTIONAL 	REQUIRED	<p>The login module's control flag. Valid values are REQUIRED, REQUISITE, SUFFICIENT, and OPTIONAL.</p> <p>SUFFICIENT This LoginModule is SUFFICIENT as per the JAAS specification. The LoginModule is not required to succeed. If authentication is successful, no other LoginModules will be called and control is returned to the caller.</p> <p>REQUISITE This LoginModule is REQUISITE as per the JAAS specification. The LoginModule is required to succeed. If authentication fails, no other LoginModules will be called and control is returned to the caller.</p> <p>REQUIRED This LoginModule is REQUIRED as per the JAAS specification. The LoginModule is required to succeed.</p> <p>OPTIONAL This LoginModule is OPTIONAL as per the JAAS specification. The LoginModule is not required to succeed.</p>
id	string		A unique configuration ID.
libraryRef	A reference to top level library element (string).		A reference to the ID of the shared library configuration.

library

A reference to the ID of the shared library configuration.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

options

A collection of JAAS Login module options

false

Java 2 Security (javaPermission)

Configuration of permissions for Java 2 Security.

Attribute name	Data type	Default value	Description
actions	string		The actions that the permission grant allows on the target name. For example, read in the case of a java.io.FilePermission.
className	string		The name of the class implementing the permission being granted. For example, java.io.FilePermission.
codebase	string		The codebase that is being granted the permission.
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
name	string		The target for which the permission applies to. For example, ALL FILES in the case of a java.io.FilePermission.
principalName	string		The principal to whom the permission is being granted.
principalType	string		The class name that would be matched for the given Principal Name.
restriction	boolean	false	Declares whether the permission is being restricted versus granted. If restriction is set to "true" then this permission is denied as opposed to being granted.

JDBC Driver (jdbcDriver)

Identifies a JDBC driver.

- library
 - file
 - fileset
 - folder

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
javax.sql.ConnectionPoolDataSource	string		JDBC driver implementation of javax.sql.ConnectionPoolDataSource.
javax.sql.DataSource	string		JDBC driver implementation of javax.sql.DataSource.
javax.sql.XADataSource	string		JDBC driver implementation of javax.sql.XADataSource.
libraryRef	A reference to top level library element (string).		Identifies JDBC driver JARs and native files.

library

Identifies JDBC driver JARs and native files.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

JNDI Entry (jndiEntry)

A single entry in the JNDI default namespace.

Attribute name	Data type	Default value	Description
decode	boolean	false	True if value needs to be decoded on lookup.
id	string		A unique configuration ID.
jndiName	string		The JNDI name to use for this entry.
value	string		The JNDI value to associate with the name.

JNDI Object Factory (jndiObjectFactory)

ObjectFactory to be used by a JNDI Reference entry.

- library
 - file
 - fileset
 - folder

Attribute name	Data type	Default value	Description
className	string		ObjectFactory implementation class name.
id	string		A unique configuration ID.
libraryRef	A reference to top level library element (string).		Library containing the factory implementation class.
objectClassName	string	java.lang.Object	Type of object returned from the factory.

library

Library containing the factory implementation class.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.

Attribute name	Data type	Default value	Description
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

JNDI Reference Entry (jndiReferenceEntry)

Reference entry in the JNDI default namespace.

- factory
 - library
 - file
 - fileset
 - folder
- properties

Attribute name	Data type	Default value	Description
decode	boolean	false	True if value needs to be decoded on lookup.
factoryRef	A reference to top level jndiObjectFactory element (string).		Object factory for the reference entry.
id	string		A unique configuration ID.
jndiName	string		JNDI name for the reference entry.

factory

Object factory for the reference entry.

false

Attribute name	Data type	Default value	Description
className	string		ObjectFactory implementation class name.
libraryRef	A reference to top level library element (string).		Library containing the factory implementation class.
objectClassName	string	java.lang.Object	Type of object returned from the factory.

factory > library

Library containing the factory implementation class.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

factory > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

factory > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

factory > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

properties

The properties for the reference entry.

false

JNDI URL Entry (jndiURLEntry)

A single entry in the JNDI default namespace that is used for binding java.net.URL entries.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
jndiName	string		The JNDI name to use for this entry.
value	string		The JNDI URL value to associate with the name.

JPA Container (jpa)

Configuration properties for the Java Persistence API container.

- excludedApplication

Attribute name	Data type	Default value	Description
defaultJtaDataSourceJndiName	string		Default Java™ Transaction API (JTA) data source JNDI name to be used by applications running in this server. By default, data sources are JTA. Only data sources that are transactional are allowed in this field.
defaultNonJtaDataSourceJndiName	string		Default non-transactional data source JNDI name to be used by applications running in this server. Only data sources that have been marked as non-transactional are allowed in this field.
defaultPersistenceProvider	string		Default persistence provider class name. If this property is not specified, the default provider is dependent on which JPA feature is enabled.
entityManagerPoolCapacity	int	-1	EntityManager pool capacity per PersistenceContext reference. The minimum is 0 and the maximum is 500.

Attribute name	Data type	Default value	Description
ignoreDataSourceErrors	boolean		If true, errors that occur while attempting to lookup a data source specified by the <jta-data-source> or <non-jta-data-source> elements in the persistence.xml file are reported and ignored, which allows the persistence provider to determine a default data source. If false, the errors are propagated to the persistence provider so that the errors can be diagnosed more easily, but misconfigured applications might not work. By default, this property is true if JPA 2.0 is enabled and false otherwise.

excludedApplication

An application to be excluded from JPA processing.

false

string

JSP Engine (jspEngine)

JSP 2.2 configuration

Attribute name	Data type	Default value	Description
disableJspRuntimeCompilation	boolean	false	Disable compilation of JSPs at runtime.
disableResourceInjection	boolean	false	Disable injection of resources into JSPs.
extendedDocumentRoot	string		Directory that the JSP engine will search for additional JSP files to serve.
jdkSourceLevel	<ul style="list-style-type: none"> • 17 • 18 • 15 • 16 • 13 • 14 	15	Default Java source level for JSPs compiled by the JSP engine. 17 17 18 18 15 15 16 16 13 13 14 14
keepGenerated	boolean	false	Keep Java source files generated for JSPs.

Attribute name	Data type	Default value	Description
prepareJSPs	int		When this attribute is present, all JSPs larger than the value (in kilobytes) are compiled at application server startup. Set this to 0 to compile all JSPs.
recompileJspOnRestart	boolean	false	Recompile JSPs after an application is restarted. JSPs are recompiled on first access.
scratchdir	string		When this attribute is set, the JSPs are compiled to the specified directory instead of the server workarea directory.
useImplicitTagLibs	boolean	true	Allow JSPs to use jsx and tsx tag libs.
useInMemory	boolean	false	Generate Java source and classes in memory (without writing to disk).

Keystore (keyStore)

A repository of security certificates used for SSL encryption.

- keyEntry

Attribute name	Data type	Default value	Description
fileBased	boolean	true	Specify true if the keystore is file based and false if the keystore is a SAF keyring or hardware keystore type.
id	string		A unique configuration ID.
location	A file, directory or url.	`\${server.output.dir}/resources/security/key.jks	An absolute or relative path to the keystore file. If a relative path is provided, the server will attempt to locate the file in the `\${server.config.dir}/resources/security` directory. Use the keystore file for a file-based keystore, the keyring name for SAF keyrings, or the device configuration file for hardware cryptography devices. In the SSL minimal configuration, the location of the file is assumed to be `\${server.config.dir}/resources/security/key.jks.

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		The password used to load the keystore file. The value can be stored in clear text or encoded form. Use the securityUtility tool to encode the password.
pollingRate	A period of time with millisecond precision	500ms	Rate at which the server checks for updates to a keystore file. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
readOnly	boolean	false	Specify true if the keystore is to be used by the server for reading and false if write operations will be performed by the server on the keystore.
type	string	jks	A keystore type supported by the target SDK.

Attribute name	Data type	Default value	Description
updateTrigger	<ul style="list-style-type: none"> • mbean • polled • disabled 	mbean	<p>Keystore file update method or trigger.</p> <p>mbean Server will only update the keystore when prompted by the FileNotificationMbean. The FileNotificationMbean is typically called by an external program such as an integrated development environment or a management application.</p> <p>polled Server will scan for keystore file changes at the polling interval and update if the keystore file has detectable changes.</p> <p>disabled Disables all update monitoring. Changes to the keystore file will not be applied while the server is running.</p>

keyEntry

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
keyPassword	Reversably encoded password (string)		Password of the private key entry in the keystore.
name	string		Name of the private key entry in the keystore.

LDAP User Registry (ldapRegistry)

Configuration properties for the LDAP user registry.

- activedFilters
- attributeConfiguration
 - attribute
 - externalIdAttribute
- contextPool

- customFilters
- domino50Filters
- edirectoryFilters
- failoverServers
 - server
- idsFilters
- iplanetFilters
- ldapCache
 - attributesCache
 - searchResultsCache
- ldapEntityType
 - objectClass
 - searchBase
- netscapeFilters
- securewayFilters

Attribute name	Data type	Default value	Description
activatedFiltersRef	A reference to top level activatedLdapFilterProperties element (string).		Specifies the list of default Microsoft Active Directory LDAP filters.
baseDN	string		Base distinguished name (DN) of the directory service, which indicates the starting point for LDAP searches in the directory service.
bindDN	string		Distinguished name (DN) for the application server, which is used to bind to the directory service.
bindPassword	Reversably encoded password (string)		Password for the bind DN. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
certificateFilter	string		Specifies the filter certificate mapping property for the LDAP filter. The filter is used to map attributes in the client certificate to entries in the LDAP registry. For example, the filter can be specified as: uid=\${SubjectCN}.

Attribute name	Data type	Default value	Description
certificateMapMode	<ul style="list-style-type: none"> EXACT_DN CERTIFICATE_FILTER 		<p>Specifies whether to map x.509 certificates into an LDAP directory by EXACT_DN or CERTIFICATE_FILTER. Specify CERTIFICATE_FILTER to use the specified certificate filter for the mapping.</p> <p>EXACT_DN exactDN</p> <p>CERTIFICATE_FILTER certFilter</p>
connectTimeout	A period of time with millisecond precision	1m	<p>Maximum time for establishing a connection to the LDAP server. An error message will be logged if the specified time expires. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.</p>
customFiltersRef	A reference to top level customLdapFilterProperties element (string).		Specifies the list of default Custom LDAP filters.
domino50FiltersRef	A reference to top level domino50LdapFilterProperties element (string).		Specifies the list of default IBM Lotus Domino LDAP filters.
edirectoryFiltersRef	A reference to top level edirectoryLdapFilterProperties element (string).		Specifies the list of Novell eDirectory LDAP filters.
host	string		Address of the LDAP server in the form of an IP address or a domain name service (DNS) name.
id	string		A unique configuration ID.
idsFiltersRef	A reference to top level idsLdapFilterProperties element (string).		Specifies the list of default IBM Tivoli Directory Server LDAP filters.
ignoreCase	boolean	true	Perform a case-insensitive authentication check.
iplanetFiltersRef	A reference to top level iplanetLdapFilterProperties element (string).		Specifies the list of default Sun Java System Directory Server LDAP filters.

Attribute name	Data type	Default value	Description
ldapType	<ul style="list-style-type: none"> • Sun Java System Directory Server • Netscape Directory Server • Microsoft Active Directory • IBM Tivoli Directory Server • IBM Lotus Domino • Custom • IBM SecureWay Directory Server • Novell eDirectory 		<p>Type of LDAP server to which a connection will be established.</p> <p>Sun Java System Directory Server iplanet</p> <p>Netscape Directory Server netscape</p> <p>Microsoft Active Directory actived</p> <p>IBM Tivoli Directory Server ibm_dir_server</p> <p>IBM Lotus Domino domino50</p> <p>Custom custom</p> <p>IBM SecureWay Directory Server secureway</p> <p>Novell eDirectory edirectory</p>
netscapeFiltersRef	A reference to top level netscapeLdapFilterProperties element (string).		Specifies the list of default Netscape Directory Server LDAP filters.
port	int		Port number of the LDAP server.
realm	string	LdapRegistry	The realm name that represents the user registry.
recursiveSearch	boolean	false	Performs a nested group search. Select this option only if the LDAP server does not support recursive server-side searches.
returnToPrimaryServer	boolean	true	A boolean value that indicates if the search should be done against the Primary Server.
reuseConnection	boolean	true	Requests the application server to reuse the LDAP server connection.

Attribute name	Data type	Default value	Description
searchTimeout	A period of time with millisecond precision	1m	Maximum time for an LDAP server to respond before a request is canceled. This is equivalent to a read timeout once the connection is established. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
securewayFiltersRef	A reference to top level securewayLdapFilterProperties element (string).		Specifies the list of default IBM SecureWay Directory Server LDAP filters.
sslEnabled	boolean	false	Indicates whether an SSL connection should be made to the LDAP server.
sslRef	A reference to top level ssl element (string).		ID of the SSL configuration to be used to connect to the SSL-enabled LDAP server.

activatedFilters

Specifies the list of default Microsoft Active Directory LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectcategory=group))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	memberOf:member	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(sAMAccountName=%v)(objectcategory=user))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	user:sAMAccountName	An LDAP filter that maps the name of a user to an LDAP entry.

attributeConfiguration

The configuration that maps the LDAP attributes with the user registry schema (for example; Person, PersonAccount or Group) field names.

false

attributeConfiguration > attribute

Define the user registry schema field names to be mapped to the LDAP attribute.

false

Attribute name	Data type	Default value	Description
defaultValue	string		The default value of the attribute.
entityType	string		The entity type of the attribute.
id	string		A unique configuration ID.
name	string		The name of the LDAP attribute.
propertyName	string		The user registry schema field name that needs to be mapped with the LDAP attribute.
syntax	string		The attribute syntax.

attributeConfiguration > externalIdAttribute

Define the name of the LDAP attribute and its properties that needs to be mapped to the user registry externalId attribute.

false

Attribute name	Data type	Default value	Description
autoGenerate	boolean	false	When enabled, the externalId attribute value is generated automatically by the user registry instead of using the value that is stored in LDAP. By default it is disabled.
entityType	string		The entity type of the attribute.
id	string		A unique configuration ID.
name	string		The name of the LDAP attribute to be used for the user registry externalId attribute.
syntax	string		The attribute syntax.

contextPool

Properties of the context pool.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	A boolean value that determines if the context pool is enabled. Disabling it can cause performance degradation.

Attribute name	Data type	Default value	Description
initialSize	int	1	An integer value that determines the initial size of the context pool. Set this based on the load on the repository.
maxSize	int	0	An integer value that defines the maximum context pool size. Set this based on the maximum load on the repository.
preferredSize	int	3	The preferred size of the context pool. Set this based on the load on the repository.
timeout	A period of time with millisecond precision	0s	The duration after which the context pool times out. An integer that represents the time that an idle context instance can remain in the pool without being closed and removed from the pool. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
waitTime	A period of time with millisecond precision	3s	The duration after which the context pool times out. The time interval that the request waits until the context pool checks again if an idle context instance is available in the pool when the number of context instances reaches the maximum pool size. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

customFilters

Specifies the list of default Custom LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)(objectclass=groupOfUniqueNames))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

domino50Filters

Specifies the list of default IBM Lotus Domino LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=dominoGroup))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	dominoGroup:member	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=Person))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	person:uid	An LDAP filter that maps the name of a user to an LDAP entry.

edirectoryFilters

Specifies the list of Novell eDirectory LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=groupOfNames))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.

Attribute name	Data type	Default value	Description
groupMemberIdMap	string	groupOfNames:member	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(cn=%v)(objectclass=Person))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	person:cn	An LDAP filter that maps the name of a user to an LDAP entry.

failoverServers

List of LDAP failover servers.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Configuration properties for LDAP failover servers. Specify it as a backup server for the primary LDAP servers. For example, <failoverServers name="failoverLdapServers"><server host="myfullyqualifiedhostname1" port="389"/><server host="myfullyqualifiedhostname2" port="389"/></failoverServers>.

failoverServers > server

Configuration properties for LDAP failover server.

false

Attribute name	Data type	Default value	Description
host	string		LDAP server host name, which can be either an IP address or a domain name service (DNS) name.
id	string		A unique configuration ID.
port	int		LDAP failover server port.

idsFilters

Specifies the list of default IBM Tivoli Directory Server LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfDynamicNames)))	An LDAP filter clause for searching the user registry for groups.

Attribute name	Data type	Default value	Description
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfUniqueNames:uniqueMember	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

iplanetFilters

Specifies the list of default Sun Java System Directory Server LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(objectclass=ldapsubentry))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	nsRole:nsRole	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=inetOrgPerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	inetOrgPerson:uid	An LDAP filter that maps the name of a user to an LDAP entry.

ldapCache

Configure the attributes of the cache.

false

ldapCache > attributesCache

The attribute cache properties configuration.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	A Boolean value to indicate that the property is enabled.

Attribute name	Data type	Default value	Description
serverTTLAttribute	string		The time after which a cache entry expires. The subsequent call for this entry will be fetched directly from the server and then placed again in the cache.
size	int	2000	Defines the number of entities that can be stored in the cache. You can increase the size of the cache based on the number of entities that are required to be stored in the cache.
sizeLimit	int	2000	The size limit for the cache.
timeout	A period of time with millisecond precision	1200ms	Defines the maximum time that the contents of the LDAP attribute cache are available. When the specified time has elapsed, the LDAP attribute cache is cleared. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

ldapCache > searchResultsCache

The configuration for the search results cache.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	A Boolean value to indicate that the property is enabled.
resultsSizeLimit	int	2000	The maximum number of results that can be returned in the search.
size	int	2000	The size of the cache. The number of search results that are stored in the cache. This needs to be configured based on the number of search queries executed on the system and the hardware system resources available.

Attribute name	Data type	Default value	Description
timeout	A period of time with millisecond precision	1200ms	Defines the maximum time that the contents of the search results cache are available. When the specified time has elapsed, the search results cache is cleared. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

ldapEntityType

Configure the LDAP object class, search filters, search bases and LDAP relative distinguished name (RDN) for Person, Group and Organizational Unit. For example, the Group entity type can have a search filter such as (&(ObjectCategory=Groupofnames)(ObjectClass=Groupofnames)) and the object class as Groupofnames with search base ou=iGroups,o=ibm,c=us.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		The name of the LDAP entity type.
searchFilter	string		A custom LDAP search expression used while searching for entity types. For example, searchFilter="(!(ObjectCategory=User)(ObjectC

ldapEntityType > objectClass

The object class defined for the given LDAP entity type in the LDAP server. For example, the object class for the group LDAP entity type can be Groupofnames.

false

string

ldapEntityType > searchBase

Specify the sub tree of the LDAP server for the search call for the given entity type which will override the base DN in search operations. For example, if the base DN is o=ibm,c=us and the search base for the PersonAccount entity type is defined to be ou=iUsers,o=ibm,c=us, then all search calls for PersonAccount will be made under subtree ou=iUsers,o=ibm,c=us. Multiple search bases can be configured for the same entity type.

false

string

netscapeFilters

Specifies the list of default Netscape Directory Server LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)))	An LDAP filter clause for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	groupOfNames:member;groupOfUniqueMember	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=inetOrgPerson)	An LDAP filter clause for searching the user registry for users.
userIdMap	string	inetOrgPerson:uid	An LDAP filter that maps the name of a user to an LDAP entry.

securewayFilters

Specifies the list of default IBM SecureWay Directory Server LDAP filters.

false

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)))	An LDAP filter clause for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	groupOfNames:member;groupOfUniqueMember	An LDAP filter that identifies user to group memberships.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

Shared Library (library)

Shared Library

- file
- fileset
- folder

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

Logging (logging)

Controls the capture and output of log and trace messages.

Attribute name	Data type	Default value	Description
consoleLogLevel	<ul style="list-style-type: none">• ERROR• WARNING• AUDIT• OFF• INFO	AUDIT	<p>The logging level used to filter messages written to system streams. The default value is audit.</p> <p>ERROR Error messages will be written to the system error stream.</p> <p>WARNING Warning messages will be written to the system output stream. Error messages will be written to the system error stream.</p> <p>AUDIT Audit and warning messages will be written to the system output stream. Error messages will be written to the system error stream.</p> <p>OFF No server output will be written to system streams. Only JVM output will be written to system streams.</p> <p>INFO Info, audit, and warning messages will be written to the system output stream. Error messages will be written to the system error stream.</p>

Attribute name	Data type	Default value	Description
copySystemStreams	boolean	true	If true, write System.out to the system output stream and System.err to the system error stream. If false, System.out and System.err will write to configured logs like messages.log or trace.log, but not to the system streams. The default value is true.
hideMessage	string		The list of messages, separated by a comma, that are configured to be hidden from the console.log and message.log files. If the messages are configured to be hidden, then they are redirected to the trace.log file.
logDirectory	Path to a directory	\${server.output.dir}/logs	Location of the directory for log files. The default value is \${server.output.dir}/logs.
maxFileSize	int Minimum: 0	20	Maximum size of a log file, in megabytes, before being rolled over; a value of 0 means no limit.
maxFiles	int Minimum: 0	2	Maximum number of log files that will be kept, before the oldest file is removed; a value of 0 means no limit.
messageFileName	string	messages.log	Name of the file to which message output will be written relative to the configured log directory. The default value is messages.log.
suppressSensitiveTrace	boolean	false	The server trace can expose sensitive data when tracing untyped data, such as bytes received over a network connection. If true, prevent potentially sensitive information from being exposed in log and trace files. The default value is false.
traceFileName	string	trace.log	Name of the file to which trace output will be written relative to the configured log directory. The default value is trace.log.

Attribute name	Data type	Default value	Description
traceFormat	<ul style="list-style-type: none"> • ENHANCED • BASIC • ADVANCED 	ENHANCED	<p>This format is used for the trace log.</p> <p>ENHANCED Use the enhanced basic trace format.</p> <p>BASIC Use the basic trace format.</p> <p>ADVANCED Use the advanced trace format.</p>
traceSpecification	string	*=info	<p>A trace specification that conforms to the trace specification grammar and specifies the initial state for various trace components. An empty value is allowed and treated as 'disable all trace'. Any component that is not specified is initialized to a default state of *=info.</p>

Logstash Collector (logstashCollector)

Logstash collector gathers data from various sources and forwards the data to a logstash server using Lumberjack protocol.

- source

Attribute name	Data type	Default value	Description
hostName	string		Host name of the logstash server.
port	int Minimum: 1 Maximum: 65535		Port number of the logstash server.
sslRef	A reference to top level ssl element (string).		Specifies an ID of the SSL repertoire that is used to connect to the logstash server.

source

Specifies a source to be used by the logstash collector.

false

string

LTPA Token (ltpa)

Lightweight Third Party Authentication (LTPA) token configuration.

Attribute name	Data type	Default value	Description
expiration	A period of time with minute precision	120m	Amount of time after which a token expires in minutes. Specify a positive integer followed by a unit of time, which can be hours (h) or minutes (m). For example, specify 30 minutes as 30m. You can include multiple values in a single entry. For example, 1h30m is equivalent to 90 minutes.
keysFileName	Path to a file	\${server.output.dir}/resources/security/ltpa.keys	Path of the file containing the token keys.
keysPassword	Reversably encoded password (string)	{xor}CDo9Hgw=	Password for the token keys. The value can be stored in clear text or encoded form. It is recommended to encode the password, use the securityUtility tool with the encode option.
monitorInterval	A period of time with millisecond precision	0ms	Rate at which the server checks for updates to the LTPA token keys file. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

Mail Session Object (mailSession)

Configuration for a Mail Session Instance.

- property

Attribute name	Data type	Default value	Description
description	string		Description of the Mail Session
from	string		The E-Mail address used to send mail with the Mail Session instance.
host	string		The host of the Mail Session
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
jndiName	string		Name of the Mail Session reference that is used for JNDI look-up
mailSessionID	string		The ID of the specific Mail Session Instance
password	Reversably encoded password (string)		The User's password, usually needed in order to connect to the host.
storeProtocol	string	imap	The Store Protocol used by the Mail Session instance. The default store protocol is IMAP
transportProtocol	string	smtp	The Transport Protocol used by the Mail Session instance. The default transport protocol is SMTP
user	string		The User's e-mail address used on the Host.

property

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		The name of the extra property
value	string		The value of the property that matches the name

Managed Executor (managedExecutorService)

Managed executor service

- contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext

Attribute name	Data type	Default value	Description
contextServiceRef	A reference to top level contextService element (string).	DefaultContextService	Configures how context is propagated to threads
id	string		A unique configuration ID.
jndiName	string		JNDI name

contextService

Configures how context is propagated to threads

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name
onError	<ul style="list-style-type: none">• IGNORE• FAIL• WARN	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

contextService > classloaderContext

Classloader context propagation configuration.

false

contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

Managed Scheduled Executor (managedScheduledExecutorService)

Managed scheduled executor service

- contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext

- classloaderContext
- jeeMetadataContext
- securityContext
- syncToOSThreadContext
- zosWLMContext

Attribute name	Data type	Default value	Description
contextServiceRef	A reference to top level contextService element (string).	DefaultContextService	Configures how context is propagated to threads
id	string		A unique configuration ID.
jndiName	string		JNDI name

contextService

Configures how context is propagated to threads

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

contextService > classloaderContext

Classloader context propagation configuration.

false

contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

Managed Thread Factory (managedThreadFactory)

Managed thread factory

- contextService
 - baseContext
 - baseContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext
 - classloaderContext
 - jeeMetadataContext
 - securityContext
 - syncToOSThreadContext
 - zosWLMContext

Attribute name	Data type	Default value	Description
contextServiceRef	A reference to top level contextService element (string).	DefaultContextService	Configures how context is propagated to threads

Attribute name	Data type	Default value	Description
createDaemonThreads	boolean	false	Configures whether or not threads created by the managed thread factory should be daemon threads.
defaultPriority	int Minimum: 1 Maximum: 10		Default priority for threads created by the managed thread factory. If unspecified, the priority of the creating thread is used. Priority cannot exceed the maximum priority for the managed thread factory, in which case the maximum priority is used instead.
id	string		A unique configuration ID.
jndiName	string		JNDI name
maxPriority	int Minimum: 1 Maximum: 10		Maximum priority for threads created by the managed thread factory.

contextService

Configures how context is propagated to threads

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

Attribute name	Data type	Default value	Description
baseContextRef	A reference to top level contextService element (string).		Specifies a base context service from which to inherit context that is not already defined on this context service.
id	string		A unique configuration ID.
jndiName	string		JNDI name

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Determines the action to take in response to configuration errors. For example, if securityContext is configured for this contextService, but the security feature is not enabled, then onError determines whether to fail, raise a warning, or ignore the parts of the configuration which are incorrect.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>

contextService > baseContext > baseContext

Specifies a base context service from which to inherit context that is not already defined on this context service.

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

Classloader context propagation configuration.

false

contextService > baseContext > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > baseContext > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > baseContext > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > baseContext > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none">• Propagate• PropagateOrNew• New	Propagate	Indicates how the WLM context should be handled for non-Daemon work. Propagate Use the same WLM Context (if one exists). PropagateOrNew Use the same WLM context or create a new one if no current context exists. New Always create a new WLM context.

contextService > classloaderContext

Classloader context propagation configuration.

false

contextService > jeeMetadataContext

Makes the namespace of the application component that submits a contextual task available to the task.

false

contextService > securityContext

When specified, the security context of the work initiator is propagated to the unit of work.

false

contextService > syncToOSThreadContext

When specified, the identity of the runAs Subject for the unit of work is synchronized with the Operating System identity.

false

contextService > zosWLMContext

Indicates that the z/OS WLM Context should be managed as part of the thread context.

false

Attribute name	Data type	Default value	Description
daemonTransactionClass	string	ASYNCDMN	The Transaction Class name provided to WLM to classify work when a new WLM context is created for Daemon work.
defaultTransactionClass	string	ASYNCBN	The Transaction Class name provided to WLM to classify work when a new WLM context is being created for non-Daemon work.
wlm	<ul style="list-style-type: none"> • Propagate • PropagateOrNew • New 	Propagate	<p>Indicates how the WLM context should be handled for non-Daemon work.</p> <p>Propagate Use the same WLM Context (if one exists).</p> <p>PropagateOrNew Use the same WLM context or create a new one if no current context exists.</p> <p>New Always create a new WLM context.</p>

Default Mime Types (mimeTypes)

Definition of mime types shared by all http virtual hosts

- type

type

Definition of mime type as id=value. Use the extension as the id, and the associated type as the value.

false

string

Monitor (monitor)

Configuration for Monitoring Feature which includes enabled traditional PMI ,FineGrained and any future configurations updates.

Attribute name	Data type	Default value	Description
enableTraditionalPMI	boolean	false	Property to enable or disable Traditional PMI way of reporting.
filter	string		Allows user to enable/disable monitors based on group name such as WebContainer,JVM,ThreadPool,Session,Connect and so on.

Netscape Directory Server LDAP Filters (netscapeLdapFilterProperties)

Specifies the list of default Netscape Directory Server LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(!(objectclass=groupOfUniqueNames)))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	groupOfNames:member;groupOfUniqueMembers	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=inetOrgPerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	inetOrgPerson:uid	An LDAP filter that maps the name of a user to an LDAP entry.

OAuth Role Map (oauth-roles)

OAuth web application security role map.

- authenticated
 - group
 - special-subject
 - user
- clientManager
 - group
 - special-subject
 - user

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

authenticated

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

authenticated > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A group access ID in the general form group:realmName/groupUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a group that has the security role.

authenticated > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		<p>One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE.</p> <p>EVERYONE All users for every request, even if the request was not authenticated.</p> <p>ALL_AUTHENTICATED_USERS All authenticated users.</p>

authenticated > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user who has the security role.

clientManager

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

clientManager > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A group access ID in the general form group:realmName/groupUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a group that has the security role.

clientManager > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		<p>One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE.</p> <p>EVERYONE All users for every request, even if the request was not authenticated.</p> <p>ALL_AUTHENTICATED_USERS All authenticated users.</p>

clientManager > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user who has the security role.

OAuth Provider Definition (oauthProvider)

OAuth provider definition.

- autoAuthorizeClient
- databaseStore
 - dataSource
 - connectionManager
 - containerAuthData
 - jaasLoginContextEntry
 - jdbcDriver
 - library
 - file
 - fileset
 - folder
 - properties
 - properties.datadirect.sqlserver
 - properties.db2.i.native
 - properties.db2.i.toolbox
 - properties.db2.jcc
 - properties.derby.client
 - properties.derby.embedded
 - properties.informix
 - properties.informix.jcc
 - properties.microsoft.sqlserver
 - properties.oracle
 - properties.sybase
 - recoveryAuthData
- grantType
- jwtGrantType
- library
 - file
 - fileset
 - folder
- localStore
 - client
 - functionalUserGroupIds
 - grantTypes
 - postLogoutRedirectUri
 - redirect
 - responseTypes
- mediatorClassname

Attribute name	Data type	Default value	Description
accessTokenLength	long	40	Length of the generated OAuth access token. The equivalent provider parameter in the full application server profile is <code>oauth20.access.token.length</code> .
accessTokenLifetime	A period of time with second precision	7200	Time that access token is valid (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.token.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
allowPublicClients	boolean	false	A value of false disables the access of public clients as detailed in the OAuth specification. The equivalent provider parameter in the full application server profile is <code>oauth20.allow.public.clients</code> .
authorizationCodeLength	long	30	Length of the generated authorization code. The equivalent provider parameter in the full application server profile is <code>oauth20.code.length</code> .
authorizationCodeLifetime	A period of time with second precision	60	Authorization code lifetime (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.code.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
authorizationErrorTemplate	string		URL of a custom authorization error page template. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.error.template</code> .
authorizationFormTemplate	string	template.html	URL of a custom authorization page template. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.form.template</code> .
authorizationGrantLifetime	A period of time with second precision	604800	Authorization grant lifetime (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.max.authorization.grant.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
autoAuthorize	boolean	false	To use auto authorization, append the <code>autoAuthorize</code> parameter to requests with a value of true. The equivalent provider parameter in the full application server profile is <code>oauth20.autoauthorize.param</code> .
autoAuthorizeParam	string	autoauthz	To use auto authorization, append the <code>autoAuthorize</code> parameter to requests with a value of true. The equivalent provider parameter in the full application server profile is <code>oauth20.autoauthorize.param</code> .
certAuthentication	boolean	false	Enable the authentication of client certificate in the https request.
characterEncoding	string		Set request character encoding to this value. The equivalent provider parameter in the full application server profile is <code>characterEncoding</code> .

Attribute name	Data type	Default value	Description
clientTokenCacheSize	long		Maximum number of entries in the client token cache.
clientURISubstitutions	string		Optional value to replace client URI strings for dynamic hostnames. The equivalent provider parameter in the full application server profile is <code>oauth20.client.uri.substitutions</code> .
consentCacheEntryLifetime	A period of time with second precision	1800	Time that an entry in the consent cache is valid (seconds). Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
consentCacheSize	long Minimum: 0	1000	Maximum number of entries allowed in the consent cache.
coverageMapSessionMaxAge	A period of time with second precision	600	The max-age value (seconds) for the cache-control header of the coverage map service. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
customLoginURL	string	login.jsp	URL of a custom login page. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.loginURL</code> .
filter	string		URI filter selects requests to be authorized by this provider. The equivalent provider parameter in the full application server profile is <code>Filter</code> .
httpsRequired	boolean	true	SSL communication between the OAuth client and provider is required.
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
includeTokenInSubject	boolean	true	If the value is true, add the com.ibm.wsspi.security.oauth2.token.WSOAuth2Token as a private credential. The equivalent provider parameter in the full application server profile is includeToken.
issueRefreshToken	boolean	true	A value of false disables generation and the use of refresh tokens. The equivalent provider parameter in the full application server profile is oauth20.issue.refresh.token.
libraryRef	A reference to top level library element (string).		Reference to shared library containing the mediator plugin class.
oauthOnly	boolean	true	If the value is true, then requests matching the filter must have an access token or they will be failed. If false, then matching requests will be checked for other authentication data if no access token is present. The equivalent provider parameter in the full application server profile is oauthOnly.
refreshTokenLength	long	50	Length of generated refresh token. The equivalent provider parameter in the full application server profile is oauth20.refresh.token.length.
skipResourceOwnerValidation	boolean	false	If the value is true, skip validation of resource owner.
userClientTokenLimit	long		Token limit for each user and client combination.

autoAuthorizeClient

Name of a client that is allowed to use auto authorization. The equivalent provider parameter in the full application server profile is oauth20.autoauthorize.clients.

false

string

databaseStore

Clients are defined and tokens are cached in the database.

false

Attribute name	Data type	Default value	Description
cleanupExpiredTokenInterval	A period of time with second precision	3600	Expired token cleanup interval (seconds). The equivalent provider parameter in the full application server profile is oauthjdbc.CleanupInterval. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
dataSourceRef	A reference to top level dataSource element (string).		Reference to the data source for the store.
password	Reversably encoded password (string)		Password used to access the database.
schema	string	OAuthDBSchema	Schema
user	string		User

databaseStore > dataSource

Reference to the data source for the store.

false

Attribute name	Data type	Default value	Description
beginTranForResultSetScrolling	boolean	true	Attempt transaction enlistment when result set scrolling interfaces are used.
beginTranForVendorAPIs	boolean	true	Attempt transaction enlistment when vendor interfaces are used.
commitOrRollbackOnCleanup	<ul style="list-style-type: none"> • commit • rollback 		<p>Determines how to clean up connections that might be in a database unit of work (AutoCommit=false) when the connection is closed or returned to the pool.</p> <p>commit Clean up the connection by committing.</p> <p>rollback Clean up the connection by rolling back.</p>
connectionManagerRef	A reference to top level connectionManager element (string).		Connection manager for a data source.

Attribute name	Data type	Default value	Description
connectionSharing	<ul style="list-style-type: none"> MatchOriginalRequest MatchCurrentState 	MatchOriginalRequest	<p>Specifies how connections are matched for sharing.</p> <p>MatchOriginalRequest When sharing connections, match based on the original connection request.</p> <p>MatchCurrentState When sharing connections, match based on the current state of the connection.</p>
containerAuthDataRef	A reference to top level authData element (string).		Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.
enableConnectionCasting	boolean	false	Indicates that connections obtained from the data source should be castable to interface classes that the JDBC vendor connection implementation implements. Enabling this option incurs additional overhead on each getConnection operation. If vendor JDBC interfaces are needed less frequently, it might be more efficient to leave this option disabled and use Connection.unwrap(interface) only where it is needed.

Attribute name	Data type	Default value	Description
isolationLevel	<ul style="list-style-type: none"> • TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT 		<p>Default transaction isolation level.</p> <p>TRANSACTION_REPEATABLE_READ Dirty reads and non-repeatable reads are prevented; phantom reads can occur.</p> <p>TRANSACTION_READ_COMMITTED Dirty reads are prevented; non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SERIALIZABLE Dirty reads, non-repeatable reads and phantom reads are prevented.</p> <p>TRANSACTION_READ_UNCOMMITTED Dirty reads, non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SNAPSHOT Snapshot isolation for Microsoft SQL Server JDBC Driver and DataDirect Connect for JDBC driver.</p>
jaasLoginContextEntryRef	A reference to top level jaasLoginContextEntry element (string).		JAAS login context entry for authentication.
jdbcDriverRef	A reference to top level jdbcDriver element (string).		JDBC driver for a data source.
jndiName	string		JNDI name for a data source.

Attribute name	Data type	Default value	Description
queryTimeout	A period of time with second precision		Default query timeout for SQL statements. In a JTA transaction, syncQueryTimeoutWithTransactionTimeout can override this default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
recoveryAuthDataRef	A reference to top level authData element (string).		Authentication data for transaction recovery.
statementCacheSize	int Minimum: 0	10	Maximum number of cached statements per connection.
supplementalJDBCTrace	boolean		Supplements the JDBC driver trace that is logged when JDBC driver trace is enabled in bootstrap.properties. JDBC driver trace specifications include: com.ibm.ws.database.logwriter, com.ibm.ws.db2.logwriter, com.ibm.ws.derby.logwriter, com.ibm.ws.informix.logwriter, com.ibm.ws.oracle.logwriter, com.ibm.ws.sqlserver.logwriter, com.ibm.ws.sybase.logwriter
syncQueryTimeoutWithTransactionTimeout	boolean	false	Use the time remaining (if any) in a JTA transaction as the default query timeout for SQL statements.
transactional	boolean	true	Enable participation in transactions that are managed by the application server.
type	<ul style="list-style-type: none"> • javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource 		Type of data source. javax.sql.DataSource javax.sql.DataSource javax.sql.XADataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource javax.sql.ConnectionPoolDataSource

databaseStore > dataSource > connectionManager

Connection manager for a data source.

false

Attribute name	Data type	Default value	Description
agedTimeout	A period of time with second precision	-1	Amount of time before a physical connection can be discarded by pool maintenance. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
connectionTimeout	A period of time with second precision	30s	Amount of time after which a connection request times out. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxConnectionsPerThread	int Minimum: 0		Limits the number of open connections on each thread.
maxIdleTime	A period of time with second precision	30m	Amount of time after which an unused or idle connection can be discarded during pool maintenance, if doing so does not reduce the pool below the minimum size. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxPoolSize	int Minimum: 0	50	Maximum number of physical connections for a pool. A value of 0 means unlimited.

Attribute name	Data type	Default value	Description
minPoolSize	int Minimum: 0		Minimum number of physical connections to maintain in the pool. The pool is not pre-populated. Aged timeout can override the minimum.
numConnectionsPerThreadLocal	int Minimum: 0		Caches the specified number of connections for each thread.
purgePolicy	<ul style="list-style-type: none"> • ValidateAllConnections • FailingConnectionOnly • EntirePool 	EntirePool	<p>Specifies which connections to destroy when a stale connection is detected in a pool.</p> <p>ValidateAllConnections When a stale connection is detected, connections are tested and those found to be bad are closed.</p> <p>FailingConnectionOnly When a stale connection is detected, only the connection which was found to be bad is closed.</p> <p>EntirePool When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed.</p>
reapTime	A period of time with second precision	3m	Amount of time between runs of the pool maintenance thread. A value of -1 disables pool maintenance. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

databaseStore > dataSource > containerAuthData

Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

databaseStore > dataSource > jaasLoginContextEntry

JAAS login context entry for authentication.

false

Attribute name	Data type	Default value	Description
loginModuleRef	List of references to top level jaasLoginModule elements (comma-separated string).	hashtable,userNameAndPassword,certificateURL	Word, certificate URL of a JAAS login module.
name	string		Name of a JAAS configuration entry.

databaseStore > dataSource > jdbcDriver

JDBC driver for a data source.

false

Attribute name	Data type	Default value	Description
javax.sql.ConnectionPoolDataSource	string		JDBC driver implementation of javax.sql.ConnectionPoolDataSource.
javax.sql.DataSource	string		JDBC driver implementation of javax.sql.DataSource.
javax.sql.XADataSource	string		JDBC driver implementation of javax.sql.XADataSource.
libraryRef	A reference to top level library element (string).		Identifies JDBC driver JARs and native files.

databaseStore > dataSource > jdbcDriver > library

Identifies JDBC driver JARs and native files.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

databaseStore > dataSource > jdbcDriver > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

databaseStore > dataSource > jdbcDriver > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

databaseStore > dataSource > jdbcDriver > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

databaseStore > dataSource > properties

List of JDBC vendor properties for the data source. For example, databaseName="dbname" serverName="localhost" portNumber="50000".

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database.
databaseName	string		JDBC driver property: databaseName.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
serverName	string		Server where the database is running.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.datadirect.sqlserver

Data source properties for the DataDirect Connect for JDBC driver for Microsoft SQL Server.

false

Attribute name	Data type	Default value	Description
JDBCBehavior	<ul style="list-style-type: none"> • 1 • 0 	0	JDBC driver property: JDBCBehavior. Values are: 0 (JDBC 4.0) or 1 (JDBC 3.0). 1 JDBC 3.0 0 JDBC 4.0
XATransactionGroup	string		JDBC driver property: XATransactionGroup.
XMLDescribeType	<ul style="list-style-type: none"> • longvarbinary • longvarchar 		JDBC driver property: XMLDescribeType. longvarbinary longvarbinary longvarchar longvarchar
accountingInfo	string		JDBC driver property: accountingInfo.
alternateServers	string		JDBC driver property: alternateServers.
alwaysReportTriggerResults	boolean		JDBC driver property: alwaysReportTriggerResults.
applicationName	string		JDBC driver property: applicationName.
authenticationMethod	<ul style="list-style-type: none"> • ntlm • userIdPassword • kerberos • auto 		JDBC driver property: authenticationMethod. ntlm ntlm userIdPassword userIdPassword kerberos kerberos auto auto
bulkLoadBatchSize	long		JDBC driver property: bulkLoadBatchSize.
bulkLoadOptions	long		JDBC driver property: bulkLoadOptions.
clientHostName	string		JDBC driver property: clientHostName.
clientUser	string		JDBC driver property: clientUser.

Attribute name	Data type	Default value	Description
codePageOverride	string		JDBC driver property: codePageOverride.
connectionRetryCount	int		JDBC driver property: connectionRetryCount.
connectionRetryDelay	A period of time with second precision		JDBC driver property: connectionRetryDelay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
convertNull	int		JDBC driver property: convertNull.
databaseName	string		JDBC driver property: databaseName.
dateTimeInputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeInputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
dateTimeOutputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeOutputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
describeInputParameters	<ul style="list-style-type: none"> • describeIfString • noDescribe • describeIfDateTime • describeAll 		JDBC driver property: describeInputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll

Attribute name	Data type	Default value	Description
describeOutputParameters	<ul style="list-style-type: none"> describeIfString noDescribe describeIfDateTime describeAll 		JDBC driver property: describeOutputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll
enableBulkLoad	boolean		JDBC driver property: enableBulkLoad.
enableCancelTimeout	boolean		JDBC driver property: enableCancelTimeout.
encryptionMethod	<ul style="list-style-type: none"> loginSSL requestSSL SSL noEncryption 		JDBC driver property: encryptionMethod. loginSSL loginSSL requestSSL requestSSL SSL SSL noEncryption noEncryption
failoverGranularity	<ul style="list-style-type: none"> disableIntegrityCheck atomicWithRepositioning nonAtomic atomic 		JDBC driver property: failoverGranularity. disableIntegrityCheck disableIntegrityCheck atomicWithRepositioning atomicWithRepositioning nonAtomic nonAtomic atomic atomic
failoverMode	<ul style="list-style-type: none"> connect select extended 		JDBC driver property: failoverMode. connect connect select select extended extended
failoverPreconnect	boolean		JDBC driver property: failoverPreconnect.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
initializationString	string		JDBC driver property: initializationString.
insensitiveResultSetBufferSize	int		JDBC driver property: insensitiveResultSetBufferSize.

Attribute name	Data type	Default value	Description
javaDoubleToString	boolean		JDBC driver property: javaDoubleToString.
loadBalancing	boolean		JDBC driver property: loadBalancing.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
longDataCacheSize	int Minimum: -1		JDBC driver property: longDataCacheSize.
netAddress	string		JDBC driver property: netAddress.
packetSize	int Minimum: -1 Maximum: 128		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
queryTimeout	A period of time with second precision		JDBC driver property: queryTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
resultsetMetaDataOptions	int		JDBC driver property: resultSetMetaDataOptions.
selectMethod	<ul style="list-style-type: none"> • direct • cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
serverName	string	localhost	Server where the database is running.
snapshotSerializable	boolean		JDBC driver property: snapshotSerializable.

Attribute name	Data type	Default value	Description
spyAttributes	string		JDBC driver property: spyAttributes.
stringInputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringInputParameterType. varchar varchar nvarchar nvarchar
stringOutputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringOutputParameterType. varchar varchar nvarchar nvarchar
suppressConnectionWarnings	boolean		JDBC driver property: suppressConnectionWarnings.
transactionMode	<ul style="list-style-type: none"> explicit implicit 		JDBC driver property: transactionMode. explicit explicit implicit implicit
truncateFractionalSeconds	boolean		JDBC driver property: truncateFractionalSeconds.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
useServerSideUpdatableCursors	boolean		JDBC driver property: useServerSideUpdatableCursors.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
validateServerCertificate	boolean		JDBC driver property: validateServerCertificate.

databaseStore > dataSource > properties.db2.i.native

Data source properties for the IBM DB2 for i Native JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> read only all read call 	all	JDBC driver property: access. read only read only all all read call read call

Attribute name	Data type	Default value	Description
autoCommit	boolean	true	JDBC driver property: autoCommit.
batchStyle	<ul style="list-style-type: none"> • 2.1 • 2.0 	2.0	JDBC driver property: batchStyle. 2.1 2.1 2.0 2.0
behaviorOverride	int		JDBC driver property: behaviorOverride.
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive
dataTruncation	string	true	JDBC driver property: dataTruncation.
databaseName	string	*LOCAL	JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy
dateSeparator	<ul style="list-style-type: none"> • \, • b • . • / • - 		JDBC driver property: dateSeparator. \, The comma character (,). b The character b . The period character (.). / The forward slash character (/). - The dash character (-).
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		JDBC driver property: decimalSeparator. \, The comma character (,). . The period character (.).
directMap	boolean	true	JDBC driver property: directMap.
doEscapeProcessing	boolean	true	JDBC driver property: doEscapeProcessing.
fullErrors	boolean		JDBC driver property: fullErrors.
libraries	string		JDBC driver property: libraries.
lobThreshold	int Maximum: 500000	0	JDBC driver property: lobThreshold.

Attribute name	Data type	Default value	Description
lockTimeout	A period of time with second precision	0	JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 63
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
networkProtocol	int		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
prefetch	boolean	true	JDBC driver property: prefetch.
queryOptimizeGoal	<ul style="list-style-type: none"> • 2 • 1 	2	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO). 2 *ALLIO 1 *FIRSTIO

Attribute name	Data type	Default value	Description
reuseObjects	boolean	true	JDBC driver property: reuseObjects.
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
systemNaming	boolean	false	JDBC driver property: systemNaming.
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • \, • b • : • . 		JDBC driver property: timeSeparator. \, The comma character (,). b The character b : The colon character (:). . The period character (.).
trace	boolean		JDBC driver property: trace.
transactionTimeout	A period of time with second precision	0	JDBC driver property: transactionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateHex	<ul style="list-style-type: none"> • binary • character 	character	JDBC driver property: translateHex. binary binary character character
useBlockInsert	boolean	false	JDBC driver property: useBlockInsert.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.db2.i.toolbox

Data source properties for the IBM DB2 for i Toolbox JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> • read only • all • read call 	all	JDBC driver property: access. read only read only all all read call read call
behaviorOverride	int		JDBC driver property: behaviorOverride.
bidiImplicitReordering	boolean	true	JDBC driver property: bidiImplicitReordering.
bidiNumericOrdering	boolean	false	JDBC driver property: bidiNumericOrdering.
bidiStringType	int		JDBC driver property: bidiStringType.
bigDecimal	boolean	true	JDBC driver property: bigDecimal.
blockCriteria	<ul style="list-style-type: none"> • 2 • 1 • 0 	2	JDBC driver property: blockCriteria. Values are: 0 (no record blocking), 1 (block if FOR FETCH ONLY is specified), 2 (block if FOR UPDATE is specified). 2 2 1 1 0 0

Attribute name	Data type	Default value	Description
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive • insensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive insensitive insensitive
dataCompression	boolean	true	JDBC driver property: dataCompression.
dataTruncation	boolean	true	JDBC driver property: dataTruncation.
databaseName	string		JDBC driver property: databaseName.
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy

Attribute name	Data type	Default value	Description
dateSeparator	<ul style="list-style-type: none"> • \, • . • / • - 		<p>JDBC driver property: dateSeparator.</p> <p>The space character ().</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p> <p>/ The forward slash character (/).</p> <p>- The dash character (-).</p>
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		<p>JDBC driver property: decimalSeparator.</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p>
driver	<ul style="list-style-type: none"> • toolbox • native 	toolbox	<p>JDBC driver property: driver.</p> <p>toolbox toolbox</p> <p>native native</p>
errors	<ul style="list-style-type: none"> • full • basic 	basic	<p>JDBC driver property: errors.</p> <p>full full</p> <p>basic basic</p>
extendedDynamic	boolean	false	JDBC driver property: extendedDynamic.
extendedMetaData	boolean	false	JDBC driver property: extendedMetaData.
fullOpen	boolean	false	JDBC driver property: fullOpen.
holdInputLocators	boolean	true	JDBC driver property: holdInputLocators.
holdStatements	boolean	false	JDBC driver property: holdStatements.
isolationLevelSwitchingSupport	boolean	false	JDBC driver property: isolationLevelSwitchingSupport.
keepAlive	boolean		JDBC driver property: keepAlive.
lazyClose	boolean	false	JDBC driver property: lazyClose.
libraries	string		JDBC driver property: libraries.

Attribute name	Data type	Default value	Description
lobThreshold	int Minimum: 0 Maximum: 16777216	0	JDBC driver property: lobThreshold.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 64
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
metaDataSource	int Minimum: 0 Maximum: 1	1	JDBC driver property: metaDataSource.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
naming	<ul style="list-style-type: none"> • system • sql 	sql	JDBC driver property: naming. system system sql sql
package	string		JDBC driver property: package.
packageAdd	boolean	true	JDBC driver property: packageAdd.
packageCCSID	<ul style="list-style-type: none"> • 13488 • 1200 	13488	JDBC driver property: packageCCSID. Values are: 1200 (UCS-2) or 13488 (UTF-16). 13488 13488 (UTF-16) 1200 1200 (UCS-2)
packageCache	boolean	false	JDBC driver property: packageCache.

Attribute name	Data type	Default value	Description
packageCriteria	<ul style="list-style-type: none"> default select 	default	JDBC driver property: packageCriteria. default default select select
packageError	<ul style="list-style-type: none"> exception none warning 	warning	JDBC driver property: packageError. exception exception none none warning warning
packageLibrary	string	QGPL	JDBC driver property: packageLibrary.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
prefetch	boolean	true	JDBC driver property: prefetch.
prompt	boolean	false	JDBC driver property: prompt.
proxyServer	string		JDBC driver property: proxyServer.
qaqqiniLibrary	string		JDBC driver property: qaqqiniLibrary.
queryOptimizeGoal	int Minimum: 0 Maximum: 2	0	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO).
receiveBufferSize	int Minimum: 1		JDBC driver property: receiveBufferSize.
remarks	<ul style="list-style-type: none"> system sql 	system	JDBC driver property: remarks. system system sql sql
rollbackCursorHold	boolean	false	JDBC driver property: rollbackCursorHold.
savePasswordWhenSerialized	boolean	false	JDBC driver property: savePasswordWhenSerialized.
secondaryUrl	string		JDBC driver property: secondaryUrl.
secure	boolean	false	JDBC driver property: secure.
sendBufferSize	int Minimum: 1		JDBC driver property: sendBufferSize.

Attribute name	Data type	Default value	Description
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
soLinger	A period of time with second precision		JDBC driver property: soLinger. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
soTimeout	A period of time with millisecond precision		JDBC driver property: soTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
sort	<ul style="list-style-type: none"> • hex • table • language 	hex	JDBC driver property: sort. hex hex table table language language
sortLanguage	string		JDBC driver property: sortLanguage.
sortTable	string		JDBC driver property: sortTable.
sortWeight	<ul style="list-style-type: none"> • unqiue • shared 		JDBC driver property: sortWeight. unqiue unique shared shared
tcpNoDelay	boolean		JDBC driver property: tcpNoDelay.
threadUsed	boolean	true	JDBC driver property: threadUsed.

Attribute name	Data type	Default value	Description
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • • \, • : • . 		JDBC driver property: timeSeparator. The space character (). \, The comma character (,). : The colon character (:). . The period character (.).
toolboxTrace	<ul style="list-style-type: none"> • diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning 		JDBC driver property: toolboxTrace. diagnostic diagnostic information information conversion conversion error error thread thread proxy proxy none none datastream datastream pcml pcml all all jdbc jdbc warning warning
trace	boolean		JDBC driver property: trace.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateBoolean	boolean	true	JDBC driver property: translateBoolean.

Attribute name	Data type	Default value	Description
translateHex	<ul style="list-style-type: none"> binary character 	character	JDBC driver property: translateHex. binary binary character character
trueAutoCommit	boolean	false	JDBC driver property: trueAutoCommit.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaLooselyCoupledSupport	int Minimum: 0 Maximum: 1	0	JDBC driver property: xaLooselyCoupledSupport.

databaseStore > dataSource > properties.db2.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for DB2.

false

Attribute name	Data type	Default value	Description
activateDatabase	int		JDBC driver property: activateDatabase.
alternateGroupDatabaseName	string		JDBC driver property: alternateGroupDatabaseName.
alternateGroupPortNumber	string		JDBC driver property: alternateGroupPortNumber.
alternateGroupServerName	string		JDBC driver property: alternateGroupServerName.
blockingReadConnectionTimeout	A period of time with second precision		JDBC driver property: blockingReadConnectionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
clientAccountingInformation	string		JDBC driver property: clientAccountingInformation.
clientApplicationInformation	string		JDBC driver property: clientApplicationInformation.
clientRerouteAlternatePortNumber	string		JDBC driver property: clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	string		JDBC driver property: clientRerouteAlternateServerName.

Attribute name	Data type	Default value	Description
clientUser	string		JDBC driver property: clientUser.
clientWorkstation	string		JDBC driver property: clientWorkstation.
connectionCloseWithInFlightTransaction	Transaction • 1		JDBC driver property: connectionCloseWithInFlightTransaction. 2 CONNECTION_CLOSE_WITH_ROLLBACK 1 CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	int		JDBC driver property: currentAlternateGroupEntry.
currentFunctionPath	string		JDBC driver property: currentFunctionPath.
currentLocaleLcCtype	string		JDBC driver property: currentLocaleLcCtype.
currentLockTimeout	A period of time with second precision		JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
currentPackagePath	string		JDBC driver property: currentPackagePath.
currentPackageSet	string		JDBC driver property: currentPackageSet.
currentSQLID	string		JDBC driver property: currentSQLID.
currentSchema	string		JDBC driver property: currentSchema.
cursorSensitivity	• 2 • 1 • 0		JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). 2 TYPE_SCROLL_ASENSITIVE 1 TYPE_SCROLL_SENSITIVE_DYNAMIC 0 TYPE_SCROLL_SENSITIVE_STATIC
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean	true	JDBC driver property: deferPrepares.

Attribute name	Data type	Default value	Description
driverType	<ul style="list-style-type: none"> 2 4 	4	JDBC driver property: driverType. 2 Type 2 JDBC driver. 4 Type 4 JDBC driver.
enableAlternateGroupSeamlessACR	boolean		JDBC driver property: enableAlternateGroupSeamlessACR.
enableClientAffinitiesList	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableClientAffinitiesList. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableExtendedDescribe	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedDescribe. 2 NO 1 YES
enableExtendedIndicators	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedIndicators. 2 NO 1 YES
enableNamedParameterMarkers	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSeamlessFailover	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeInputStreams	boolean		JDBC driver property: fullyMaterializeInputStreams.
fullyMaterializeInputStreamsOnBatchExecution	<ul style="list-style-type: none"> 1 		JDBC driver property: fullyMaterializeInputStreamsOnBatchExecution. 2 NO 1 YES
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.

Attribute name	Data type	Default value	Description
implicitRollbackOption	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: implicitRollbackOption.</p> <p>2 IMPLICIT_ROLLBACK_OPTION_N 1 IMPLICIT_ROLLBACK_OPTION_N 0 IMPLICIT_ROLLBACK_OPTION_N</p>
interruptProcessingMode	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: interruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODI 1 INTERRUPT_PROCESSING_MODI 0 INTERRUPT_PROCESSING_MODI</p>
keepAliveTimeOut	A period of time with second precision		<p>JDBC driver property: keepAliveTimeOut. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
keepDynamic	int		JDBC driver property: keepDynamic.
kerberosServerPrincipal	string		JDBC driver property: kerberosServerPrincipal.
loginTimeout	A period of time with second precision		<p>JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
maxConnCachedParamBufferSize	int		JDBC driver property: maxConnCachedParamBufferSize.
maxRetriesForClientReroute	int		JDBC driver property: maxRetriesForClientReroute.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	50000	Port on which to obtain database connections.
profileName	string		JDBC driver property: profileName.

Attribute name	Data type	Default value	Description
queryCloseImplicit	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryCloseImplicit. Values are: 1 (QUERY_CLOSE_IMPLICIT_YES) or 2 (QUERY_CLOSE_IMPLICIT_NO).</p> <p>2 QUERY_CLOSE_IMPLICIT_NO 1 QUERY_CLOSE_IMPLICIT_YES</p>
queryDataSize	<p>int</p> <p>Minimum: 4096</p> <p>Maximum: 65535</p>		JDBC driver property: queryDataSize.
queryTimeoutInterruptProcessingMode	<p>int</p> <ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryTimeoutInterruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODE_C 1 INTERRUPT_PROCESSING_MODE_S</p>
readOnly	boolean		JDBC driver property: readOnly.
recordTemporalHistory	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: recordTemporalHistory.</p> <p>2 NO 1 YES</p>
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.

Attribute name	Data type	Default value	Description
retryIntervalForClientReroute	A period of time with second precision		JDBC driver property: retryIntervalForClientReroute. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12 		<p>JDBC driver property: securityMechanism. Values are:</p> <ul style="list-style-type: none"> 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY), 11 (KERBEROS_SECURITY), 12 (ENCRYPTED_USER_AND_DATA_SECURITY), 13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY), 15 (PLUGIN_SECURITY), 16 (ENCRYPTED_USER_ONLY_SECURITY), 18 (TLS_CLIENT_CERTIFICATE_SECURITY). <p>3 CLEAR_TEXT_PASSWORD_SECURITY</p> <p>7 ENCRYPTED_PASSWORD_SECURITY</p> <p>4 USER_ONLY_SECURITY</p> <p>18 TLS_CLIENT_CERTIFICATE_SECURITY</p> <p>15 PLUGIN_SECURITY</p> <p>9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p> <p>16 ENCRYPTED_USER_ONLY_SECURITY</p> <p>13 ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY</p> <p>11 KERBEROS_SECURITY</p> <p>12 ENCRYPTED_USER_AND_DATA_SECURITY</p>
sendDataAsIs	boolean		JDBC driver property: sendDataAsIs.
serverName	string	localhost	Server where the database is running.
sessionTimeZone	string		JDBC driver property: sessionTimeZone.
sqljCloseStmtsWithOpenResultSet	boolean		JDBC driver property: sqljCloseStmtsWithOpenResultSet.

Attribute name	Data type	Default value	Description
sqljEnableClassLoaderSpecificProfiles	boolean		JDBC driver property: sqljEnableClassLoaderSpecificProfiles.
sslConnection	boolean		JDBC driver property: sslConnection.
streamBufferSize	int		JDBC driver property: streamBufferSize.
stripTrailingZerosForDecimalNumbers	int • 1		JDBC driver property: stripTrailingZerosForDecimalNumbers. 2 NO 1 YES
sysSchema	string		JDBC driver property: sysSchema.
timerLevelForQueryTimeOut	int • 2 • 1 • -1		JDBC driver property: timerLevelForQueryTimeOut. 2 QUERYTIMEOUT_CONNECTION_LEVEL 1 QUERYTIMEOUT_STATEMENT_LEVEL -1 QUERYTIMEOUT_DISABLED
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceFileCount	int		JDBC driver property: traceFileCount.
traceFileSize	int		JDBC driver property: traceFileSize.
traceLevel	int	0	Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
traceOption	int • 1 • 0		JDBC driver property: traceOption 1 1 0 0

Attribute name	Data type	Default value	Description
translateForBitData	<ul style="list-style-type: none"> 2 1 		JDBC driver property: translateForBitData. 2 SERVER_ENCODING_REPRESENTATION 1 HEX_REPRESENTATION
updateCountForBatch	<ul style="list-style-type: none"> 2 1 		JDBC driver property: updateCountForBatch. 2 TOTAL_UPDATE_COUNT 1 NO_UPDATE_COUNT
useCachedCursor	boolean		JDBC driver property: useCachedCursor.
useIdentityValLocalForAutoGeneratedKeys	boolean		JDBC driver property: useIdentityValLocalForAutoGeneratedKeys.
useJDBC41DefinitionForGetColumns	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC41DefinitionForGetColumns. 2 NO 1 YES
useJDBC4ColumnNameAndLabelSemantics	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
useTransactionRedirect	boolean		JDBC driver property: useTransactionRedirect.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaNetworkOptimization	boolean		JDBC driver property: xaNetworkOptimization.

databaseStore > dataSource > properties.derby.client

Data source properties for Derby Network Client JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1527	Port on which to obtain database connections.
retrieveMessageText	boolean	true	JDBC driver property: retrieveMessageText.
securityMechanism	<ul style="list-style-type: none"> 3 7 4 9 8 	3	<p>JDBC driver property: securityMechanism. Values are:</p> <p>3 (CLEAR_TEXT_PASSWORD_SECURITY),</p> <p>4 (USER_ONLY_SECURITY),</p> <p>7 (ENCRYPTED_PASSWORD_SECURITY),</p> <p>8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY),</p> <p>9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY</p> <p>7 ENCRYPTED_PASSWORD_SECURITY</p> <p>4 USER_ONLY_SECURITY</p> <p>9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p> <p>8 STRONG_PASSWORD_SUBSTITUTE_SECURITY</p>

Attribute name	Data type	Default value	Description
serverName	string	localhost	Server where the database is running.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		JDBC driver property: shutdownDatabase. false Do not shut down the database. shutdown Shut down the database when a connection is attempted.
ssl	<ul style="list-style-type: none"> basic off peerAuthentication 		JDBC driver property: ssl. basic basic off off peerAuthentication peerAuthentication
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_XA_CALLS=2048, TRACE_ALL=-1.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.derby.embedded
 Data source properties for Derby Embedded JDBC driver.
 false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		<p>JDBC driver property: shutdownDatabase.</p> <p>false Do not shut down the database.</p> <p>shutdown Shut down the database when a connection is attempted.</p>
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.informix
 Data source properties for the Informix JDBC driver.
 false

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
ifxCLIENT_LOCALE	string		JDBC driver property: ifxCLIENT_LOCALE.
ifxCPMAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMInitPoolSize	int		JDBC driver property: ifxCPMInitPoolSize.
ifxCPMMaxConnections	int		JDBC driver property: ifxCPMMaxConnections.
ifxCPMMaxPoolSize	int		JDBC driver property: ifxCPMMaxPoolSize.
ifxCPMMinAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMMinAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMMinPoolSize	int		JDBC driver property: ifxCPMMinPoolSize.
ifxCPMServiceInterval	A period of time with millisecond precision		JDBC driver property: ifxCPMServiceInterval. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxDBANSIWARN	boolean		JDBC driver property: ifxDBANSIWARN.
ifxDBCENTURY	string		JDBC driver property: ifxDBCENTURY.
ifxDBDATE	string		JDBC driver property: ifxDBDATE.
ifxDBSPACETEMP	string		JDBC driver property: ifxDBSPACETEMP.

Attribute name	Data type	Default value	Description
ifxDBTEMP	string		JDBC driver property: ifxDBTEMP.
ifxDBTIME	string		JDBC driver property: ifxDBTIME.
ifxDBUPSPACE	string		JDBC driver property: ifxDBUPSPACE.
ifxDB_LOCALE	string		JDBC driver property: ifxDB_LOCALE.
ifxDELIMIDENT	boolean		JDBC driver property: ifxDELIMIDENT.
ifxENABLE_TYPE_CACHE	boolean		JDBC driver property: ifxENABLE_TYPE_CACHE.
ifxFET_BUF_SIZE	int		JDBC driver property: ifxFET_BUF_SIZE.
ifxGL_DATE	string		JDBC driver property: ifxGL_DATE.
ifxGL_DATETIME	string		JDBC driver property: ifxGL_DATETIME.
ifxIFXHOST	string	localhost	JDBC driver property: ifxIFXHOST.
ifxIFX_AUTOFREE	boolean		JDBC driver property: ifxIFX_AUTOFREE.
ifxIFX_DIRECTIVES	string		JDBC driver property: ifxIFX_DIRECTIVES.
ifxIFX_LOCK_MODE_WAIT	A period of time with second precision	2s	JDBC driver property: ifxIFX_LOCK_MODE_WAIT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxIFX_SOC_TIMEOUT	A period of time with millisecond precision		JDBC driver property: ifxIFX_SOC_TIMEOUT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxIFX_USEPUT	boolean		JDBC driver property: ifxIFX_USEPUT.

Attribute name	Data type	Default value	Description
ifxIFX_USE_STRENC	boolean		JDBC driver property: ifxIFX_USE_STRENC.
ifxIFX_XASPEC	string	y	JDBC driver property: ifxIFX_XASPEC.
ifxINFORMIXCONRETRY	int		JDBC driver property: ifxINFORMIXCONRETRY.
ifxINFORMIXCONTIME	A period of time with second precision		JDBC driver property: ifxINFORMIXCONTIME. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxINFORMIXOPCACHE	string		JDBC driver property: ifxINFORMIXOPCACHE.
ifxINFORMIXSTACKSIZE	int		JDBC driver property: ifxINFORMIXSTACKSIZE.
ifxJDBCTEMP	string		JDBC driver property: ifxJDBCTEMP.
ifxLDAP_IFXBASE	string		JDBC driver property: ifxLDAP_IFXBASE.
ifxLDAP_PASSWD	string		JDBC driver property: ifxLDAP_PASSWD.
ifxLDAP_URL	string		JDBC driver property: ifxLDAP_URL.
ifxLDAP_USER	string		JDBC driver property: ifxLDAP_USER.
ifxLOBCACHE	int		JDBC driver property: ifxLOBCACHE.
ifxNEWCODESET	string		JDBC driver property: ifxNEWCODESET.
ifxNEWLOCALE	string		JDBC driver property: ifxNEWLOCALE.
ifxNODEFDAC	string		JDBC driver property: ifxNODEFDAC.
ifxOPTCOMPIND	string		JDBC driver property: ifxOPTCOMPIND.
ifxOPTOFC	string		JDBC driver property: ifxOPTOFC.
ifxOPT_GOAL	string		JDBC driver property: ifxOPT_GOAL.
ifxPATH	string		JDBC driver property: ifxPATH.
ifxPDQPRIORITY	string		JDBC driver property: ifxPDQPRIORITY.

Attribute name	Data type	Default value	Description
ifxPLCONFIG	string		JDBC driver property: ifxPLCONFIG.
ifxPLOAD_LO_PATH	string		JDBC driver property: ifxPLOAD_LO_PATH.
ifxPROTOCOLTRACE	int		JDBC driver property: ifxPROTOCOLTRACE.
ifxPROTOCOLTRACEFILE	string		JDBC driver property: ifxPROTOCOLTRACEFILE.
ifxPROXY	string		JDBC driver property: ifxPROXY.
ifxPSORT_DBTEMP	string		JDBC driver property: ifxPSORT_DBTEMP.
ifxPSORT_NPROCS	boolean		JDBC driver property: ifxPSORT_NPROCS.
ifxSECURITY	string		JDBC driver property: ifxSECURITY.
ifxSQLH_FILE	string		JDBC driver property: ifxSQLH_FILE.
ifxSQLH_LOC	string		JDBC driver property: ifxSQLH_LOC.
ifxSQLH_TYPE	string		JDBC driver property: ifxSQLH_TYPE.
ifxSSLCONNECTION	string		JDBC driver property: ifxSSLCONNECTION.
ifxSTMT_CACHE	string		JDBC driver property: ifxSTMT_CACHE.
ifxTRACE	int		JDBC driver property: ifxTRACE.
ifxTRACEFILE	string		JDBC driver property: ifxTRACEFILE.
ifxTRUSTED_CONTEXT	string		JDBC driver property: ifxTRUSTED_CONTEXT.
ifxUSEV5SERVER	boolean		JDBC driver property: ifxUSEV5SERVER.
ifxUSE_DTENV	boolean		JDBC driver property: ifxUSE_DTENV.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
roleName	string		JDBC driver property: roleName.
serverName	string		Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.informix.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for Informix.

false

Attribute name	Data type	Default value	Description
DBANSIWARN	boolean		JDBC driver property: DBANSIWARN.
DBDATE	string		JDBC driver property: DBDATE.
DBPATH	string		JDBC driver property: DBPATH.
DBSPACETEMP	string		JDBC driver property: DBSPACETEMP.
DBTEMP	string		JDBC driver property: DBTEMP.
DBUPSPACE	string		JDBC driver property: DBUPSPACE.
DELIMIDENT	boolean		JDBC driver property: DELIMIDENT.
IFX_DIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_DIRECTIVES. ON ON OFF OFF
IFX_EXTDIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_EXTDIRECTIVES. ON ON OFF OFF
IFX_UPDESC	string		JDBC driver property: IFX_UPDESC.

Attribute name	Data type	Default value	Description
IFX_XASTDCOMPLIANCE_XAEND	<ul style="list-style-type: none"> 0 		JDBC driver property: IFX_XASTDCOMPLIANCE_XAEND. 1 1 0 0
INFORMIXOPCACHE	string		JDBC driver property: INFORMIXOPCACHE.
INFORMIXSTACKSIZE	string		JDBC driver property: INFORMIXSTACKSIZE.
NODEFDAC	<ul style="list-style-type: none"> yes no 		JDBC driver property: NODEFDAC. yes yes no no
OPTCOMPIND	<ul style="list-style-type: none"> 2 1 0 		JDBC driver property: OPTCOMPIND. 2 2 1 1 0 0
OPTOFC	<ul style="list-style-type: none"> 1 0 		JDBC driver property: OPTOFC. 1 1 0 0
PDQPRIORITY	<ul style="list-style-type: none"> HIGH LOW OFF 		JDBC driver property: PDQPRIORITY. HIGH HIGH LOW LOW OFF OFF
PSORT_DBTEMP	string		JDBC driver property: PSORT_DBTEMP.
PSORT_NPROCS	string Maximum: 10		JDBC driver property: PSORT_NPROCS.
STMT_CACHE	<ul style="list-style-type: none"> 1 0 		JDBC driver property: STMT_CACHE. 1 1 0 0
currentLockTimeout	A period of time with second precision	2s	JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean		JDBC driver property: deferPrepares.
driverType	int	4	JDBC driver property: driverType.
enableNamedParameterMarkers	int		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO).
enableSeamlessFailover	int		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO).
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.
keepDynamic	int		JDBC driver property: keepDynamic.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
progressiveStreaming	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: progressiveStreaming. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
queryDataSize	int Minimum: 4096 Maximum: 10485760		JDBC driver property: queryDataSize.

Attribute name	Data type	Default value	Description
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 9 		<p>JDBC driver property: securityMechanism. Values are: 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY).</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p>
serverName	string	localhost	Server where the database is running.
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.

Attribute name	Data type	Default value	Description
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
useJDBC4ColumnNameAndLabelSemantics	boolean		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO).
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.microsoft.sqlserver

Data source properties for Microsoft SQL Server JDBC Driver.

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database. Example: jdbc:sqlserver://localhost:1433;databaseName=myDB.
applicationIntent	<ul style="list-style-type: none"> ReadOnly ReadWrite 		JDBC driver property: applicationIntent. ReadOnly ReadOnly ReadWrite ReadWrite
applicationName	string		JDBC driver property: applicationName.
authenticationScheme	<ul style="list-style-type: none"> NativeAuthentication JavaKerberos 		JDBC driver property: authenticationScheme. NativeAuthentication NativeAuthentication JavaKerberos JavaKerberos

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
encrypt	boolean		JDBC driver property: encrypt.
failoverPartner	string		JDBC driver property: failoverPartner.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
instanceName	string		JDBC driver property: instanceName.
integratedSecurity	boolean		JDBC driver property: integratedSecurity.
lastUpdateCount	boolean		JDBC driver property: lastUpdateCount.
lockTimeout	A period of time with millisecond precision		JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
multiSubnetFailover	boolean		JDBC driver property: multiSubnetFailover.
packetSize	int Minimum: 512 Maximum: 32767		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.

Attribute name	Data type	Default value	Description
responseBuffering	<ul style="list-style-type: none"> full adaptive 		JDBC driver property: responseBuffering. full full adaptive adaptive
selectMethod	<ul style="list-style-type: none"> direct cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
sendStringParametersAsUnicode	boolean	false	JDBC driver property: sendStringParametersAsUnicode.
sendTimeAsDatetime	boolean		JDBC driver property: sendTimeAsDatetime.
serverName	string	localhost	Server where the database is running.
trustServerCertificate	boolean		JDBC driver property: trustServerCertificate.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
workstationID	string		JDBC driver property: workstationID.
xopenStates	boolean		JDBC driver property: xopenStates.

databaseStore > dataSource > properties.oracle

Data source properties for Oracle JDBC driver.

false

Attribute name	Data type	Default value	Description
ONSConfiguration	string		JDBC driver property: ONSConfiguration.
TNSEntryName	string		JDBC driver property: TNSEntryName.
URL	string		URL for connecting to the database. Examples: jdbc:oracle:thin://localhost:1521/sample or jdbc:oracle:oci://localhost:1521/sample.
connectionProperties	string		JDBC driver property: connectionProperties.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
driverType	<ul style="list-style-type: none"> oci thin 	thin	JDBC driver property: driverType. oci oci thin thin
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	string		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1521	Port on which to obtain database connections.
serverName	string	localhost	Server where the database is running.
serviceName	string		JDBC driver property: serviceName.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

databaseStore > dataSource > properties.sybase

Data source properties for Sybase JDBC driver.

false

Attribute name	Data type	Default value	Description
SERVER_INITIATED_TRANSACTIONS	<ul style="list-style-type: none"> false true 	false	JDBC driver property: SERVER_INITIATED_TRANSACTIONS. false false true true
connectionProperties	string	SELECT_OPENS_CURSOR=true	JDBC driver property: connectionProperties.
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	<ul style="list-style-type: none"> • SSL • socket 		JDBC driver property: networkProtocol. SSL SSL socket socket
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	5000	Port on which to obtain database connections.
resourceManagerName	string		JDBC driver property: resourceManagerName.
serverName	string	localhost	Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
version	int		JDBC driver property: version.

databaseStore > dataSource > recoveryAuthData

Authentication data for transaction recovery.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

grantType

An access token grant type (as detailed in the OAuth specification) that is allowed for the provider. The equivalent provider parameter in the full application server profile is `oauth20.grant.types.allowed`.

false

string

jwtGrantType

The `grant_type` for JWT Token handler

false

Attribute name	Data type	Default value	Description
clockSkew	A period of time with second precision	300s	The time difference allowed between OpenID Connect Client and OpenID Connect Provider systems when they are not synchronized. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
iatRequired	boolean	false	The iat claim in a jwt token is required.
maxJtiCacheSize	long Minimum: 1	10000	The maximum size of cache, which keeps jti data of jwt token, to prevent the jti from being reused.
tokenMaxLifetime	A period of time with second precision	7200s	The time indicates the maximum lifetime of an alive jwt token since its issued-at-time. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

library

Reference to shared library containing the mediator plugin class.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

localStore

Clients are defined in server.xml and tokens are cached in the server.

false

Attribute name	Data type	Default value	Description
tokenStoreSize	long	2000	Token store size

localStore > client

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
applicationType	<ul style="list-style-type: none"> • native • web 	web	The type of application best describing the client. native native web web
displayname	string		Display name of the client.
enabled	boolean	true	Client is enabled if true, disabled if false.

Attribute name	Data type	Default value	Description
functionalUserId	string		A user identifier to be associated with access tokens obtained by this client using the client credentials grant type. When this client parameter is specified, the value is returned in the functional_user_id response parameter from the introspect endpoint.
id	string		A unique configuration ID.
introspectTokens	boolean	false	Boolean value specifying whether the client is allowed to access the introspection endpoint to introspect tokens issued by the authorization server.
name	string		Name of the client (sometimes referred to as the Id).
preAuthorizedScope	string		Space separated list of scope values that the client can use when requesting access tokens that are deemed to have been pre-approved by the resource owner and therefore does not require the resource owner's consent.
scope	string		Specify by spaces the list of scopes of the client.
secret	Reversably encoded password (string)		Secret key of the client.
sessionManaged	boolean	false	Boolean indicating whether the client participates in OpenID session management.
subjectType	<ul style="list-style-type: none"> public 		Subject type requested for response to this client. public public
tokenEndpointAuthMethod	<ul style="list-style-type: none"> client_secret_post none client_secret_basic 	client_secret_basic	The requested authentication method for the token endpoint of the client. client_secret_post client_secret_post none none client_secret_basic client_secret_basic

localStore > client > functionalUserGroupIds

A list of group ids to be associated with access tokens obtained by this client using the client credentials grant type. When this client parameter is specified, the value is returned in the functional_user_groupIds response parameter from the introspect endpoint.

false

string

localStore > client > grantTypes

Grant types the client may use.

false

localStore > client > postLogoutRedirectUris

Array of URLs supplied by the RP to which it may request that the end-user's user agent be redirected using the post_logout_redirect_uri parameter after a logout has been performed.

false

string

localStore > client > redirect

Array of redirect URIs for use in redirect-based flows such as the authorization code and implicit grant types of the client. The first redirect URI is used as a default, when none is specified in a request.

false

string

localStore > client > responseTypes

Response types the client may use.

false

mediatorClassname

Mediator plugin class name. The equivalent provider parameter in the full application server profile is oauth20.mediator.classnames.

false

string

OpenId Authentication (openid)

A variety of custom properties are available for OpenId authentication.

- authFilter
 - host
 - remoteAddress
 - requestUrl
 - userAgent
 - webApp
- userInfo

Attribute name	Data type	Default value	Description
authFilterRef	A reference to top level authFilter element (string).		Specifies the authentication filter reference.

Attribute name	Data type	Default value	Description
authenticationMode	<ul style="list-style-type: none"> • checkid_immediate • checkid_setup 	checkid_setup	<p>Specifies the OpenID provider authentication mode either checkid_immediate or checkid_setup. checkid_setup is the default authentication mode.</p> <p>checkid_immediate The checkid_immediate disables the browser interact with the user.</p> <p>checkid_setup The checkid_setup enables the openID provider to interact with the user, to request authentication or self-registration before returning a result to the openId relying party.</p>
hashAlgorithm	<ul style="list-style-type: none"> • SHA256 • SHA1 	SHA256	<p>Specifies the hash algorithm that is used to sign and encrypt the OpenID provider response parameters.</p> <p>SHA256 Secure hash algorithm SHA256</p> <p>SHA1 Secure hash algorithm SHA1</p>
hostNameVerificationEnabled	boolean	true	Specifies whether enable host name verification or not.
httpsRequired	boolean	true	Require SSL communication between the OpenID relying party and provider service.
mapIdentityToRegistryUser	boolean	false	Specifies whether to map identity to registry user. The user registry is not used to create the user subject.
providerIdentifier	string		Specifies a default OpenID provider URL where users get the Open IDs.
realmIdentifier	string		Specifies the attribute for the OpenID provider name.

Attribute name	Data type	Default value	Description
sslRef	A reference to top level ssl element (string).		Specifies an ID of the SSL configuration is used to connect to the OpenID provider.
useClientIdentity	boolean	false	Specifies whether to use the client OpenID identity to create a user subject. If set to true, only the OpenID client identity is used. If set to false and the first element of userInfoRef is found, we use it to create a user subject. Otherwise, we use the OpenID identity to create a user subject.
userInfoRef	List of references to top level userInfo elements (comma-separated string).	email	Specifies a list of userInfo references separated by commas for the OpenID provider to include in the response.

authFilter

Specifies the authentication filter reference.

false

authFilter > host

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

authFilter > remoteAddress

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ip	string		Specifies the IP address.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> lessThan equals greaterThan contains notContain 	contains	Specifies the match type. lessThan Less than equals Equals greaterThan Greater than contains Contains notContain Not contain

authFilter > requestUrl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> equals contains notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
urlPattern	string		Specifies the URL pattern.

authFilter > userAgent

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
agent	string		Specifies the user agent
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> equals contains notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain

authFilter > webApp

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

userInfo

Specifies a list of userInfo references separated by commas for the OpenID provider to include in the response.

false

Attribute name	Data type	Default value	Description
alias	string	email	Specifies an alias name.
count	int Minimum: 1	1	Specifies how much userInfo is included in the response of the openID provider.
id	string		A unique configuration ID.
required	boolean	true	Specifies whether user information is required or not.
uriType	string	http://axschema.org/contact/email	Specifies a URI type.

OpenID Connect Client (openidConnectClient)

OpenID Connect client.

- audiences
- authFilter
 - host
 - remoteAddress
 - requestUrl
 - userAgent
 - webApp

Attribute name	Data type	Default value	Description
authFilterRef	A reference to top level authFilter element (string).		Specifies the authentication filter reference.
authnSessionDisabled	boolean	true	An authentication session cookie will not be created for inbound propagation. The client is expected to send a valid OAuth token for every request.
authorizationEndpointUrl	string		Specifies an Authorization end point URL.

Attribute name	Data type	Default value	Description
clientId	string		Identity of the client.
clientSecret	Reversably encoded password (string)		Secret key of the client.
createSession	boolean	true	Specifies whether to create an HttpSession if the current HttpSession does not exist.
disableIssChecking	boolean	false	Do not check for the issuer while validating the json response for inbound token propagation.
disableLtpaCookie	boolean	false	Do not create an LTPA Token during processing of the OAuth token. Create a cookie of the specific Service Provider instead.
grantType	<ul style="list-style-type: none"> • implicit • authorization_code 	authorization_code	<p>Specifies the grant type to use for this client.</p> <p>implicit Implicit grant type</p> <p>authorization_code Authorization code grant type</p>
groupIdIdentifier	string	groupIds	Specifies a JSON attribute in the ID token that is used as the name of the group that the authenticated principal is a member of.
headerName	string		The name of the header which carries the inbound token in the request.
hostNameVerificationEnabled	boolean	false	Specifies whether to enable host name verification.
httpsRequired	boolean	true	Require SSL communication between the OpenID relying party and provider service.
id	string		A unique configuration ID.
inboundPropagation	<ul style="list-style-type: none"> • supported • none • required 	none	<p>Controls the operation of the token inbound propagation of the OpenID relying party.</p> <p>supported Support inbound token propagation</p> <p>none Do not support inbound token propagation</p> <p>required Require inbound token propagation</p>

Attribute name	Data type	Default value	Description
includeIdTokenInSubject	boolean	true	Specifies whether to include ID token in the client subject.
initialStateCacheCapacity	int Minimum: 0	3000	Specifies the beginning capacity of state cache. The capacity grows bigger when needed by itself.
isClientSideRedirectSupported	boolean	true	Specifies whether the client supports redirect at client side.
issuerIdentifier	string		An Issuer Identifier is a case-sensitive URL using the HTTPS scheme that contains scheme, host and optionally port number and path components.
jwkEndpointUrl	string		Specifies a JWK end point URL.
mapIdentityToRegistryUser	boolean	false	Specifies whether to map the identity to a registry user. If this is set to false, then the user registry is not used to create the user subject.
nonceEnabled	boolean	false	Enable the nonce parameter in the authorization code flow.
reAuthnCushion	A period of time with millisecond precision	0s	The time period to authenticate a user again when its tokens are about to expire. The expiration time of an ID token is specified by its exp claim. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
reAuthnOnAccessTokenExpiration	boolean	true	Authenticate a user again when its authenticating access token expires and disableLtpaCookie is set to true.
realmIdentifier	string	realmName	Specifies a JSON attribute in the ID token that is used as the realm name.

Attribute name	Data type	Default value	Description
realmName	string		Specifies a realm name to be used to create the user subject when the mapIdentityToRegistryUser is set to false.
redirectToRPHostAndPort	string		Specifies a redirect OpenID relying party host and port number.
scope	tokenType	openid profile	OpenID Connect scope (as detailed in the OpenID Connect specification) that is allowed for the provider.
signatureAlgorithm	<ul style="list-style-type: none"> • HS256 • none • RS256 	HS256	<p>Specifies the signature algorithm that will be used to verify the signature of the ID token.</p> <p>HS256 Use the HS256 signature algorithm to sign and verify tokens</p> <p>none Tokens are not required to be signed</p> <p>RS256 Use the RS256 signature algorithm to sign and verify tokens</p>
sslRef	A reference to top level ssl element (string).		Specifies an ID of the SSL configuration that is used to connect to the OpenID Connect provider.
tokenEndpointAuthMethod	<ul style="list-style-type: none"> • post • basic 	post	<p>The method to use for sending credentials to the token endpoint of the OpenID Connect provider in order to authenticate the client.</p> <p>post post</p> <p>basic basic</p>
tokenEndpointUrl	string		Specifies a token end point URL.
trustAliasName	string		Key alias name to locate public key for signature validation with asymmetric algorithm.
trustStoreRef	A reference to top level keyStore element (string).		A keystore containing the public key necessary for verifying the signature of the ID token.

Attribute name	Data type	Default value	Description
uniqueUserIdentifier	string	uniqueSecurityName	Specifies a JSON attribute in the ID token that is used as the unique user name as it applies to the WSCredential in the subject.
userIdentifier	string		Specifies a JSON attribute in the ID token that is used as the user principal name in the subject. If no value is specified, the JSON attribute "sub" is used.
userIdentityToCreateSubject	string	sub	Specifies a user identity in the ID token used to create the user subject.
validationEndpointUrl	string		The endpoint URL for validating the token inbound propagation. The type of endpoint is decided by the validationMethod.
validationMethod	<ul style="list-style-type: none"> • introspect • userinfo 	introspect	<p>The method of validation on the token inbound propagation.</p> <p>introspect Validate inbound tokens using token introspection</p> <p>userinfo Validate inbound tokens using the userinfo end point</p>

audiences

The trusted audience list that is verified against the aud claim in the JSON web token.

false

string

authFilter

Specifies the authentication filter reference.

false

authFilter > host

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

authFilter > remoteAddress

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ip	string		Specifies the IP address.
matchType	<ul style="list-style-type: none"> • lessThan • equals • greaterThan • contains • notContain 	contains	Specifies the match type. lessThan Less than equals Equals greaterThan Greater than contains Contains notContain Not contain

authFilter > requestUrl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
urlPattern	string		Specifies the URL pattern.

authFilter > userAgent

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
agent	string		Specifies the user agent
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain

authFilter > webApp

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

OpenID Connect Server Provider (openidConnectProvider)

OpenID Connect server provider

- claimToUserRegistryMap
 - property
- customClaims
- discovery
 - claimsSupported
 - grantTypesSupported
 - idTokenSigningAlgValuesSupported
 - responseModesSupported
 - responseTypesSupported
 - scopesSupported
 - tokenEndpointAuthMethodsSupported
- oauthProvider
 - autoAuthorizeClient
 - databaseStore
 - dataSource
 - connectionManager
 - containerAuthData
 - jaasLoginContextEntry
 - jdbcDriver

- library
 - file
 - fileset
 - folder
- properties
- properties.datadirect.sqlserver
- properties.db2.i.native
- properties.db2.i.toolbox
- properties.db2.jcc
- properties.derby.client
- properties.derby.embedded
- properties.informix
- properties.informix.jcc
- properties.microsoft.sqlserver
- properties.oracle
- properties.sybase
- recoveryAuthData
- grantType
- jwtGrantType
- library
 - file
 - fileset
 - folder
- localStore
 - client
 - functionalUserGroupIds
 - grantTypes
 - postLogoutRedirectUris
 - redirect
 - responseTypes
- mediatorClassname
- scopeToClaimMap
 - property

Attribute name	Data type	Default value	Description
allowDefaultSsoCookieName	boolean	false	When this property is set to true, the default SSO cookie name, ltpaToken2, is used if a custom SSO cookie name is not configured. If a custom cookie name is configured for SSO, that cookie name is used. If a custom cookie name is not configured and this property is set to false, an auto-generated SSO cookie name will be used.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
idTokenLifetime	A period of time with second precision	2h	Time that ID token is valid (seconds). Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
issuerIdentifier	string		Specify an issuer identifier for the issuer of the response.
jwkEnabled	boolean	false	Enables or disables JWK.
jwkRotationTime	A period of time with minute precision	720m	Amount of time after which a new JWK will be generated. Specify a positive integer followed by a unit of time, which can be hours (h) or minutes (m). For example, specify 30 minutes as 30m. You can include multiple values in a single entry. For example, 1h30m is equivalent to 90 minutes.
jwkSigningKeySize	<ul style="list-style-type: none"> • 4096 • 1024 • 2048 	2048	Size measured in bits of the signing key. 4096 4096 bits 1024 1024 bits 2048 2048 bits
keyAliasName	string		Key alias name to locate the private key for signing with an asymmetric algorithm.
keyStoreRef	A reference to top level keyStore element (string).	opKeyStore	A keystore containing the private key necessary for signing with an asymmetric algorithm.
oauthProviderRef	A reference to top level oauthProvider element (string).		A reference to the ID of an OAuth provider.
sessionManaged	boolean	false	Indicate by true or false whether session management is supported. Default is false.

Attribute name	Data type	Default value	Description
signatureAlgorithm	<ul style="list-style-type: none"> • HS256 • none • RS256 	HS256	<p>Specify the signature algorithm that will be used to sign the ID token.</p> <p>HS256 HMAC using SHA-256 hash</p> <p>none No signature</p> <p>RS256 RSASSA-PKCS-v1_5 using SHA-256 hash</p>
trustStoreRef	A reference to top level keyStore element (string).		A keystore containing the public key necessary for verifying a signature of the JWT token.

claimToUserRegistryMap

Specify the user registry key for the claim.

false

Attribute name	Data type	Default value	Description
address	string	postalAddress	Specify the user registry key that will be retrieved for the address claim.
email	string	mail	Specify the user registry key that will be retrieved for the email claim.
given_name	string	givenName	Specify the user registry key that will be retrieved for the given_name claim.
name	string	displayName	Specify the user registry key that will be retrieved for the name claim.
phone_number	string	telephoneNumber	Specify the user registry key that will be retrieved for the phone_number claim.
picture	string	photoURL	Specify the user registry key that will be retrieved for the picture claim.

claimToUserRegistryMap > property

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Specify the name of the property
value	string		Specify the value of the property

customClaims

The extra claims to be put in the payloads of the ID Token, in addition to the default realmName, uniqueSecurityName and groupIds claims.

false

string

discovery

Discovery is based on OpenID Connect and Jazz Authorization Server Profile.

false

Attribute name	Data type	Default value	Description
claimsParameterSupported	boolean	false	Indicate by true or false whether claims parameter is supported.
requestParameterSupported	boolean	false	Indicate by true or false whether request parameter is supported.
requestUriParameterSupported	boolean	false	Indicate by true or false whether request URI parameter is supported.
requireRequestUriRegistration	boolean	false	Indicate by true or false whether require request URI registration is supported.

discovery > claimsSupported

Specify by comma the list of claims that will be supported.

false

string

discovery > grantTypesSupported

Specify by comma the list of the grant types that will be used.

false

discovery > idTokenSigningAlgValuesSupported

Specify the signature algorithm that will be used to sign the ID token.

false

discovery > responseModesSupported

Specify by comma the list of the response modes that will be used.

false

discovery > responseTypesSupported

Specify by comma the list of the response types that will be supported by the OP.

false

discovery > scopesSupported

Specify by comma the list of scopes that will be supported.

false

string

discovery > tokenEndpointAuthMethodsSupported

Specify by comma the list of the token endpoint authentication methods that will be used.

false

oauthProvider

A reference to the ID of an OAuth provider.

false

Attribute name	Data type	Default value	Description
accessTokenLength	long	40	Length of the generated OAuth access token. The equivalent provider parameter in the full application server profile is <code>oauth20.access.token.length</code> .
accessTokenLifetime	A period of time with second precision	7200	Time that access token is valid (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.token.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
allowPublicClients	boolean	false	A value of false disables the access of public clients as detailed in the OAuth specification. The equivalent provider parameter in the full application server profile is <code>oauth20.allow.public.clients</code> .
authorizationCodeLength	long	30	Length of the generated authorization code. The equivalent provider parameter in the full application server profile is <code>oauth20.code.length</code> .

Attribute name	Data type	Default value	Description
authorizationCodeLifetime	A period of time with second precision	60	Authorization code lifetime (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.code.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
authorizationErrorTemplate	string		URL of a custom authorization error page template. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.error.template</code> .
authorizationFormTemplate	string	template.html	URL of a custom authorization page template. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.form.template</code> .
authorizationGrantLifetime	A period of time with second precision	604800	Authorization grant lifetime (seconds). The equivalent provider parameter in the full application server profile is <code>oauth20.max.authorization.grant.lifetime.seconds</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
autoAuthorize	boolean	false	To use auto authorization, append the <code>autoAuthorize</code> parameter to requests with a value of true. The equivalent provider parameter in the full application server profile is <code>oauth20.autoauthorize.param</code> .

Attribute name	Data type	Default value	Description
autoAuthorizeParam	string	autoauthz	To use auto authorization, append the autoAuthorize parameter to requests with a value of true. The equivalent provider parameter in the full application server profile is oauth20.autoauthorize.param.
certAuthentication	boolean	false	Enable the authentication of client certificate in the https request.
characterEncoding	string		Set request character encoding to this value. The equivalent provider parameter in the full application server profile is characterEncoding.
clientTokenCacheSize	long		Maximum number of entries in the client token cache.
clientURISubstitutions	string		Optional value to replace client URI strings for dynamic hostnames. The equivalent provider parameter in the full application server profile is oauth20.client.uri.substitutions.
consentCacheEntryLifetime	A period of time with second precision	1800	Time that an entry in the consent cache is valid (seconds). Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
consentCacheSize	long Minimum: 0	1000	Maximum number of entries allowed in the consent cache.
coverageMapSessionMaxAge	A period of time with second precision	600	The max-age value (seconds) for the cache-control header of the coverage map service. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
customLoginURL	string	login.jsp	URL of a custom login page. The equivalent provider parameter in the full application server profile is <code>oauth20.authorization.loginURL</code> .
filter	string		URI filter selects requests to be authorized by this provider. The equivalent provider parameter in the full application server profile is <code>Filter</code> .
httpsRequired	boolean	true	SSL communication between the OAuth client and provider is required.
includeTokenInSubject	boolean	true	If the value is true, add the <code>com.ibm.wsspi.security.oauth20.token.WSOAuth</code> as a private credential. The equivalent provider parameter in the full application server profile is <code>includeToken</code> .
issueRefreshToken	boolean	true	A value of false disables generation and the use of refresh tokens. The equivalent provider parameter in the full application server profile is <code>oauth20.issue.refresh.token</code> .
libraryRef	A reference to top level library element (string).		Reference to shared library containing the mediator plugin class.
oauthOnly	boolean	true	If the value is true, then requests matching the filter must have an access token or they will be failed. If false, then matching requests will be checked for other authentication data if no access token is present. The equivalent provider parameter in the full application server profile is <code>oauthOnly</code> .
refreshTokenLength	long	50	Length of generated refresh token. The equivalent provider parameter in the full application server profile is <code>oauth20.refresh.token.length</code> .
skipResourceOwnerValidation	boolean	false	If the value is true, skip validation of resource owner.
userClientTokenLimit	long		Token limit for each user and client combination.

oauthProvider > autoAuthorizeClient

Name of a client that is allowed to use auto authorization. The equivalent provider parameter in the full application server profile is `oauth20.autoauthorize.clients`.

false

string

oauthProvider > databaseStore

Clients are defined and tokens are cached in the database.

false

Attribute name	Data type	Default value	Description
<code>cleanupExpiredTokenInterval</code>	A period of time with second precision	3600	Expired token cleanup interval (seconds). The equivalent provider parameter in the full application server profile is <code>oauthjdbc.CleanupInterval</code> . Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
<code>dataSourceRef</code>	A reference to top level <code>dataSource</code> element (string).		Reference to the data source for the store.
<code>password</code>	Reversably encoded password (string)		Password used to access the database.
<code>schema</code>	string	<code>OAuthDBSchema</code>	Schema
<code>user</code>	string		User

oauthProvider > databaseStore > dataSource

Reference to the data source for the store.

false

Attribute name	Data type	Default value	Description
<code>beginTranForResultSetScrollingApps</code>	boolean	true	Attempt transaction enlistment when result set scrolling interfaces are used.
<code>beginTranForVendorAPIs</code>	boolean	true	Attempt transaction enlistment when vendor interfaces are used.

Attribute name	Data type	Default value	Description
commitOrRollbackOnCleanup	<ul style="list-style-type: none"> • commit • rollback 		<p>Determines how to clean up connections that might be in a database unit of work (AutoCommit=false) when the connection is closed or returned to the pool.</p> <p>commit Clean up the connection by committing.</p> <p>rollback Clean up the connection by rolling back.</p>
connectionManagerRef	A reference to top level connectionManager element (string).		Connection manager for a data source.
connectionSharing	<ul style="list-style-type: none"> • MatchOriginalRequest • MatchCurrentState 	MatchOriginalRequest	<p>Specifies how connections are matched for sharing.</p> <p>MatchOriginalRequest When sharing connections, match based on the original connection request.</p> <p>MatchCurrentState When sharing connections, match based on the current state of the connection.</p>
containerAuthDataRef	A reference to top level authData element (string).		Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

Attribute name	Data type	Default value	Description
enableConnectionCasting	boolean	false	Indicates that connections obtained from the data source should be castable to interface classes that the JDBC vendor connection implementation implements. Enabling this option incurs additional overhead on each getConnection operation. If vendor JDBC interfaces are needed less frequently, it might be more efficient to leave this option disabled and use Connection.unwrap(interface) only where it is needed.
isolationLevel	<ul style="list-style-type: none"> • TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT 		<p>Default transaction isolation level.</p> <p>TRANSACTION_REPEATABLE_READ Dirty reads and non-repeatable reads are prevented; phantom reads can occur.</p> <p>TRANSACTION_READ_COMMITTED Dirty reads are prevented; non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SERIALIZABLE Dirty reads, non-repeatable reads and phantom reads are prevented.</p> <p>TRANSACTION_READ_UNCOMMITTED Dirty reads, non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SNAPSHOT Snapshot isolation for Microsoft SQL Server JDBC Driver and DataDirect Connect for JDBC driver.</p>

Attribute name	Data type	Default value	Description
jaasLoginContextEntryRef	A reference to top level jaasLoginContextEntry element (string).		JAAS login context entry for authentication.
jdbcDriverRef	A reference to top level jdbcDriver element (string).		JDBC driver for a data source.
jndiName	string		JNDI name for a data source.
queryTimeout	A period of time with second precision		Default query timeout for SQL statements. In a JTA transaction, syncQueryTimeoutWithTransactionTimeout can override this default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
recoveryAuthDataRef	A reference to top level authData element (string).		Authentication data for transaction recovery.
statementCacheSize	int Minimum: 0	10	Maximum number of cached statements per connection.
supplementalJDBCTrace	boolean		Supplements the JDBC driver trace that is logged when JDBC driver trace is enabled in bootstrap.properties. JDBC driver trace specifications include: com.ibm.ws.database.logwriter, com.ibm.ws.db2.logwriter, com.ibm.ws.derby.logwriter, com.ibm.ws.informix.logwriter, com.ibm.ws.oracle.logwriter, com.ibm.ws.sqlserver.logwriter, com.ibm.ws.sybase.logwriter
syncQueryTimeoutWithTransactionTimeout	boolean	false	Use the time remaining (if any) in a JTA transaction as the default query timeout for SQL statements.
transactional	boolean	true	Enable participation in transactions that are managed by the application server.

Attribute name	Data type	Default value	Description
type	<ul style="list-style-type: none"> javax.sql.DataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource 		Type of data source. javax.sql.DataSource javax.sql.DataSource javax.sql.XADataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource javax.sql.ConnectionPoolDataSource

oauthProvider > databaseStore > dataSource > connectionManager

Connection manager for a data source.

false

Attribute name	Data type	Default value	Description
agedTimeout	A period of time with second precision	-1	Amount of time before a physical connection can be discarded by pool maintenance. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
connectionTimeout	A period of time with second precision	30s	Amount of time after which a connection request times out. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxConnectionsPerThread	int Minimum: 0		Limits the number of open connections on each thread.

Attribute name	Data type	Default value	Description
maxIdleTime	A period of time with second precision	30m	Amount of time after which an unused or idle connection can be discarded during pool maintenance, if doing so does not reduce the pool below the minimum size. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxPoolSize	int Minimum: 0	50	Maximum number of physical connections for a pool. A value of 0 means unlimited.
minPoolSize	int Minimum: 0		Minimum number of physical connections to maintain in the pool. The pool is not pre-populated. Aged timeout can override the minimum.
numConnectionsPerThreadLocal	int Minimum: 0		Caches the specified number of connections for each thread.

Attribute name	Data type	Default value	Description
purgePolicy	<ul style="list-style-type: none"> • ValidateAllConnections • FailingConnectionOnly • EntirePool 	EntirePool	<p>Specifies which connections to destroy when a stale connection is detected in a pool.</p> <p>ValidateAllConnections When a stale connection is detected, connections are tested and those found to be bad are closed.</p> <p>FailingConnectionOnly When a stale connection is detected, only the connection which was found to be bad is closed.</p> <p>EntirePool When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed.</p>
reapTime	A period of time with second precision	3m	Amount of time between runs of the pool maintenance thread. A value of -1 disables pool maintenance. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

oauthProvider > databaseStore > dataSource > containerAuthData

Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

oauthProvider > databaseStore > dataSource > jaasLoginContextEntry

JAAS login context entry for authentication.

false

Attribute name	Data type	Default value	Description
loginModuleRef	List of references to top level jaasLoginModule elements (comma-separated string).	hashtable,userNameAndPassword,certificateURL	Word certificateURL of a JAAS login module.
name	string		Name of a JAAS configuration entry.

oauthProvider > databaseStore > dataSource > jdbcDriver

JDBC driver for a data source.

false

Attribute name	Data type	Default value	Description
javax.sql.ConnectionPoolDataSource	string		JDBC driver implementation of javax.sql.ConnectionPoolDataSource.
javax.sql.DataSource	string		JDBC driver implementation of javax.sql.DataSource.
javax.sql.XADataSource	string		JDBC driver implementation of javax.sql.XADataSource.
libraryRef	A reference to top level library element (string).		Identifies JDBC driver JARs and native files.

oauthProvider > databaseStore > dataSource > jdbcDriver > library

Identifies JDBC driver JARs and native files.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

oauthProvider > databaseStore > dataSource > jdbcDriver > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

oauthProvider > databaseStore > dataSource > jdbcDriver > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

oauthProvider > databaseStore > dataSource > jdbcDriver > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

oauthProvider > databaseStore > dataSource > properties

List of JDBC vendor properties for the data source. For example, databaseName="dbname" serverName="localhost" portNumber="50000".

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database.
databaseName	string		JDBC driver property: databaseName.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
serverName	string		Server where the database is running.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.datadirect.sqlserver

Data source properties for the DataDirect Connect for JDBC driver for Microsoft SQL Server.

false

Attribute name	Data type	Default value	Description
JDBCBehavior	<ul style="list-style-type: none"> • 1 • 0 	0	JDBC driver property: JDBCBehavior. Values are: 0 (JDBC 4.0) or 1 (JDBC 3.0). 1 JDBC 3.0 0 JDBC 4.0
XATransactionGroup	string		JDBC driver property: XATransactionGroup.
XMLDescribeType	<ul style="list-style-type: none"> • longvarbinary • longvarchar 		JDBC driver property: XMLDescribeType. longvarbinary longvarbinary longvarchar longvarchar
accountingInfo	string		JDBC driver property: accountingInfo.
alternateServers	string		JDBC driver property: alternateServers.
alwaysReportTriggerResults	boolean		JDBC driver property: alwaysReportTriggerResults.
applicationName	string		JDBC driver property: applicationName.
authenticationMethod	<ul style="list-style-type: none"> • ntlm • userIdPassword • kerberos • auto 		JDBC driver property: authenticationMethod. ntlm ntlm userIdPassword userIdPassword kerberos kerberos auto auto
bulkLoadBatchSize	long		JDBC driver property: bulkLoadBatchSize.
bulkLoadOptions	long		JDBC driver property: bulkLoadOptions.
clientHostName	string		JDBC driver property: clientHostName.

Attribute name	Data type	Default value	Description
clientUser	string		JDBC driver property: clientUser.
codePageOverride	string		JDBC driver property: codePageOverride.
connectionRetryCount	int		JDBC driver property: connectionRetryCount.
connectionRetryDelay	A period of time with second precision		JDBC driver property: connectionRetryDelay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
convertNull	int		JDBC driver property: convertNull.
databaseName	string		JDBC driver property: databaseName.
dateTimeInputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeInputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
dateTimeOutputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeOutputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
describeInputParameters	<ul style="list-style-type: none"> • describeIfString • noDescribe • describeIfDateTime • describeAll 		JDBC driver property: describeInputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll

Attribute name	Data type	Default value	Description
describeOutputParameters	<ul style="list-style-type: none"> describeIfString noDescribe describeIfDateTime describeAll 		JDBC driver property: describeOutputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll
enableBulkLoad	boolean		JDBC driver property: enableBulkLoad.
enableCancelTimeout	boolean		JDBC driver property: enableCancelTimeout.
encryptionMethod	<ul style="list-style-type: none"> loginSSL requestSSL SSL noEncryption 		JDBC driver property: encryptionMethod. loginSSL loginSSL requestSSL requestSSL SSL SSL noEncryption noEncryption
failoverGranularity	<ul style="list-style-type: none"> disableIntegrityCheck atomicWithRepositioning nonAtomic atomic 		JDBC driver property: failoverGranularity. disableIntegrityCheck disableIntegrityCheck atomicWithRepositioning atomicWithRepositioning nonAtomic nonAtomic atomic atomic
failoverMode	<ul style="list-style-type: none"> connect select extended 		JDBC driver property: failoverMode. connect connect select select extended extended
failoverPreconnect	boolean		JDBC driver property: failoverPreconnect.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
initializationString	string		JDBC driver property: initializationString.
insensitiveResultSetBufferSize	int		JDBC driver property: insensitiveResultSetBufferSize.

Attribute name	Data type	Default value	Description
javaDoubleToString	boolean		JDBC driver property: javaDoubleToString.
loadBalancing	boolean		JDBC driver property: loadBalancing.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
longDataCacheSize	int Minimum: -1		JDBC driver property: longDataCacheSize.
netAddress	string		JDBC driver property: netAddress.
packetSize	int Minimum: -1 Maximum: 128		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
queryTimeout	A period of time with second precision		JDBC driver property: queryTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
resultsetMetaDataOptions	int		JDBC driver property: resultSetMetaDataOptions.
selectMethod	<ul style="list-style-type: none"> • direct • cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
serverName	string	localhost	Server where the database is running.
snapshotSerializable	boolean		JDBC driver property: snapshotSerializable.

Attribute name	Data type	Default value	Description
spyAttributes	string		JDBC driver property: spyAttributes.
stringInputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringInputParameterType. varchar varchar nvarchar nvarchar
stringOutputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringOutputParameterType. varchar varchar nvarchar nvarchar
suppressConnectionWarnings	boolean		JDBC driver property: suppressConnectionWarnings.
transactionMode	<ul style="list-style-type: none"> explicit implicit 		JDBC driver property: transactionMode. explicit explicit implicit implicit
truncateFractionalSeconds	boolean		JDBC driver property: truncateFractionalSeconds.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
useServerSideUpdatableCursors	boolean		JDBC driver property: useServerSideUpdatableCursors.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
validateServerCertificate	boolean		JDBC driver property: validateServerCertificate.

oauthProvider > databaseStore > dataSource > properties.db2.i.native

Data source properties for the IBM DB2 for i Native JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> read only all read call 	all	JDBC driver property: access. read only read only all all read call read call

Attribute name	Data type	Default value	Description
autoCommit	boolean	true	JDBC driver property: autoCommit.
batchStyle	<ul style="list-style-type: none"> • 2.1 • 2.0 	2.0	JDBC driver property: batchStyle. 2.1 2.1 2.0 2.0
behaviorOverride	int		JDBC driver property: behaviorOverride.
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive
dataTruncation	string	true	JDBC driver property: dataTruncation.
databaseName	string	*LOCAL	JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy
dateSeparator	<ul style="list-style-type: none"> • \, • b • . • / • - 		JDBC driver property: dateSeparator. \, The comma character (,). b The character b . The period character (.). / The forward slash character (/). - The dash character (-).
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		JDBC driver property: decimalSeparator. \, The comma character (,). . The period character (.).
directMap	boolean	true	JDBC driver property: directMap.
doEscapeProcessing	boolean	true	JDBC driver property: doEscapeProcessing.
fullErrors	boolean		JDBC driver property: fullErrors.
libraries	string		JDBC driver property: libraries.
lobThreshold	int Maximum: 500000	0	JDBC driver property: lobThreshold.

Attribute name	Data type	Default value	Description
lockTimeout	A period of time with second precision	0	JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 63
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
networkProtocol	int		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
prefetch	boolean	true	JDBC driver property: prefetch.
queryOptimizeGoal	<ul style="list-style-type: none"> • 2 • 1 	2	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO). 2 *ALLIO 1 *FIRSTIO

Attribute name	Data type	Default value	Description
reuseObjects	boolean	true	JDBC driver property: reuseObjects.
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
systemNaming	boolean	false	JDBC driver property: systemNaming.
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • \, • b • : • . 		JDBC driver property: timeSeparator. \, The comma character (,). b The character b : The colon character (:). . The period character (.).
trace	boolean		JDBC driver property: trace.
transactionTimeout	A period of time with second precision	0	JDBC driver property: transactionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateHex	<ul style="list-style-type: none"> • binary • character 	character	JDBC driver property: translateHex. binary binary character character
useBlockInsert	boolean	false	JDBC driver property: useBlockInsert.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.db2.i.toolbox

Data source properties for the IBM DB2 for i Toolbox JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> • read only • all • read call 	all	JDBC driver property: access. read only read only all all read call read call
behaviorOverride	int		JDBC driver property: behaviorOverride.
bidiImplicitReordering	boolean	true	JDBC driver property: bidiImplicitReordering.
bidiNumericOrdering	boolean	false	JDBC driver property: bidiNumericOrdering.
bidiStringType	int		JDBC driver property: bidiStringType.
bigDecimal	boolean	true	JDBC driver property: bigDecimal.
blockCriteria	<ul style="list-style-type: none"> • 2 • 1 • 0 	2	JDBC driver property: blockCriteria. Values are: 0 (no record blocking), 1 (block if FOR FETCH ONLY is specified), 2 (block if FOR UPDATE is specified). 2 2 1 1 0 0

Attribute name	Data type	Default value	Description
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive • insensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive insensitive insensitive
dataCompression	boolean	true	JDBC driver property: dataCompression.
dataTruncation	boolean	true	JDBC driver property: dataTruncation.
databaseName	string		JDBC driver property: databaseName.
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy

Attribute name	Data type	Default value	Description
dateSeparator	<ul style="list-style-type: none"> • • \, • . • / • - 		<p>JDBC driver property: dateSeparator.</p> <p>The space character ().</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p> <p>/ The forward slash character (/).</p> <p>- The dash character (-).</p>
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		<p>JDBC driver property: decimalSeparator.</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p>
driver	<ul style="list-style-type: none"> • toolbox • native 	toolbox	<p>JDBC driver property: driver.</p> <p>toolbox toolbox</p> <p>native native</p>
errors	<ul style="list-style-type: none"> • full • basic 	basic	<p>JDBC driver property: errors.</p> <p>full full</p> <p>basic basic</p>
extendedDynamic	boolean	false	JDBC driver property: extendedDynamic.
extendedMetaData	boolean	false	JDBC driver property: extendedMetaData.
fullOpen	boolean	false	JDBC driver property: fullOpen.
holdInputLocators	boolean	true	JDBC driver property: holdInputLocators.
holdStatements	boolean	false	JDBC driver property: holdStatements.
isolationLevelSwitchingSupport	boolean	false	JDBC driver property: isolationLevelSwitchingSupport.
keepAlive	boolean		JDBC driver property: keepAlive.
lazyClose	boolean	false	JDBC driver property: lazyClose.
libraries	string		JDBC driver property: libraries.

Attribute name	Data type	Default value	Description
lobThreshold	int Minimum: 0 Maximum: 16777216	0	JDBC driver property: lobThreshold.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 64
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
metaDataSource	int Minimum: 0 Maximum: 1	1	JDBC driver property: metaDataSource.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
naming	<ul style="list-style-type: none"> • system • sql 	sql	JDBC driver property: naming. system system sql sql
package	string		JDBC driver property: package.
packageAdd	boolean	true	JDBC driver property: packageAdd.
packageCCSID	<ul style="list-style-type: none"> • 13488 • 1200 	13488	JDBC driver property: packageCCSID. Values are: 1200 (UCS-2) or 13488 (UTF-16). 13488 13488 (UTF-16) 1200 1200 (UCS-2)
packageCache	boolean	false	JDBC driver property: packageCache.

Attribute name	Data type	Default value	Description
packageCriteria	<ul style="list-style-type: none"> default select 	default	JDBC driver property: packageCriteria. default default select select
packageError	<ul style="list-style-type: none"> exception none warning 	warning	JDBC driver property: packageError. exception exception none none warning warning
packageLibrary	string	QGPL	JDBC driver property: packageLibrary.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
prefetch	boolean	true	JDBC driver property: prefetch.
prompt	boolean	false	JDBC driver property: prompt.
proxyServer	string		JDBC driver property: proxyServer.
qaqqiniLibrary	string		JDBC driver property: qaqqiniLibrary.
queryOptimizeGoal	int Minimum: 0 Maximum: 2	0	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO).
receiveBufferSize	int Minimum: 1		JDBC driver property: receiveBufferSize.
remarks	<ul style="list-style-type: none"> system sql 	system	JDBC driver property: remarks. system system sql sql
rollbackCursorHold	boolean	false	JDBC driver property: rollbackCursorHold.
savePasswordWhenSerialized	boolean	false	JDBC driver property: savePasswordWhenSerialized.
secondaryUrl	string		JDBC driver property: secondaryUrl.
secure	boolean	false	JDBC driver property: secure.
sendBufferSize	int Minimum: 1		JDBC driver property: sendBufferSize.

Attribute name	Data type	Default value	Description
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
soLinger	A period of time with second precision		JDBC driver property: soLinger. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
soTimeout	A period of time with millisecond precision		JDBC driver property: soTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
sort	<ul style="list-style-type: none"> • hex • table • language 	hex	JDBC driver property: sort. hex hex table table language language
sortLanguage	string		JDBC driver property: sortLanguage.
sortTable	string		JDBC driver property: sortTable.
sortWeight	<ul style="list-style-type: none"> • unqiue • shared 		JDBC driver property: sortWeight. unqiue unique shared shared
tcpNoDelay	boolean		JDBC driver property: tcpNoDelay.
threadUsed	boolean	true	JDBC driver property: threadUsed.

Attribute name	Data type	Default value	Description
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • • \, • : • . 		JDBC driver property: timeSeparator. The space character (). \, The comma character (,). : The colon character (:). . The period character (.).
toolboxTrace	<ul style="list-style-type: none"> • diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning 		JDBC driver property: toolboxTrace. diagnostic diagnostic information information conversion conversion error error thread thread proxy proxy none none datastream datastream pcml pcml all all jdbc jdbc warning warning
trace	boolean		JDBC driver property: trace.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateBoolean	boolean	true	JDBC driver property: translateBoolean.

Attribute name	Data type	Default value	Description
translateHex	<ul style="list-style-type: none"> binary character 	character	JDBC driver property: translateHex. binary binary character character
trueAutoCommit	boolean	false	JDBC driver property: trueAutoCommit.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaLooselyCoupledSupport	int Minimum: 0 Maximum: 1	0	JDBC driver property: xaLooselyCoupledSupport.

oauthProvider > databaseStore > dataSource > properties.db2.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for DB2.

false

Attribute name	Data type	Default value	Description
activateDatabase	int		JDBC driver property: activateDatabase.
alternateGroupDatabaseName	string		JDBC driver property: alternateGroupDatabaseName.
alternateGroupPortNumber	string		JDBC driver property: alternateGroupPortNumber.
alternateGroupServerName	string		JDBC driver property: alternateGroupServerName.
blockingReadConnectionTimeout	A period of time with second precision		JDBC driver property: blockingReadConnectionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
clientAccountingInformation	string		JDBC driver property: clientAccountingInformation.
clientApplicationInformation	string		JDBC driver property: clientApplicationInformation.
clientRerouteAlternatePortNumber	string		JDBC driver property: clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	string		JDBC driver property: clientRerouteAlternateServerName.

Attribute name	Data type	Default value	Description
clientUser	string		JDBC driver property: clientUser.
clientWorkstation	string		JDBC driver property: clientWorkstation.
connectionCloseWithInFlightTransaction	Transaction • 1		JDBC driver property: connectionCloseWithInFlightTransaction. 2 CONNECTION_CLOSE_WITH_ROLLBACK 1 CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	int		JDBC driver property: currentAlternateGroupEntry.
currentFunctionPath	string		JDBC driver property: currentFunctionPath.
currentLocaleLcCtype	string		JDBC driver property: currentLocaleLcCtype.
currentLockTimeout	A period of time with second precision		JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
currentPackagePath	string		JDBC driver property: currentPackagePath.
currentPackageSet	string		JDBC driver property: currentPackageSet.
currentSQLID	string		JDBC driver property: currentSQLID.
currentSchema	string		JDBC driver property: currentSchema.
cursorSensitivity	• 2 • 1 • 0		JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). 2 TYPE_SCROLL_ASENSITIVE 1 TYPE_SCROLL_SENSITIVE_DYNAMIC 0 TYPE_SCROLL_SENSITIVE_STATIC
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean	true	JDBC driver property: deferPrepares.

Attribute name	Data type	Default value	Description
driverType	<ul style="list-style-type: none"> 2 4 	4	JDBC driver property: driverType. 2 Type 2 JDBC driver. 4 Type 4 JDBC driver.
enableAlternateGroupSeamlessACR	boolean		JDBC driver property: enableAlternateGroupSeamlessACR.
enableClientAffinitiesList	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableClientAffinitiesList. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableExtendedDescribe	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedDescribe. 2 NO 1 YES
enableExtendedIndicators	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedIndicators. 2 NO 1 YES
enableNamedParameterMarkers	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSeamlessFailover	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeInputStreams	boolean		JDBC driver property: fullyMaterializeInputStreams.
fullyMaterializeInputStreamsOnBatchExecution	<ul style="list-style-type: none"> 1 		JDBC driver property: fullyMaterializeInputStreamsOnBatchExecution. 2 NO 1 YES
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.

Attribute name	Data type	Default value	Description
implicitRollbackOption	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: implicitRollbackOption.</p> <p>2 IMPLICIT_ROLLBACK_OPTION_CLOSE</p> <p>1 IMPLICIT_ROLLBACK_OPTION_NO</p> <p>0 IMPLICIT_ROLLBACK_OPTION_NO</p>
interruptProcessingMode	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: interruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODE_C</p> <p>1 INTERRUPT_PROCESSING_MODE_S</p> <p>0 INTERRUPT_PROCESSING_MODE_D</p>
keepAliveTimeOut	A period of time with second precision		<p>JDBC driver property: keepAliveTimeOut. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
keepDynamic	int		JDBC driver property: keepDynamic.
kerberosServerPrincipal	string		JDBC driver property: kerberosServerPrincipal.
loginTimeout	A period of time with second precision		<p>JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
maxConnCachedParamBufferSize	int		JDBC driver property: maxConnCachedParamBufferSize.
maxRetriesForClientReroute	int		JDBC driver property: maxRetriesForClientReroute.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	50000	Port on which to obtain database connections.
profileName	string		JDBC driver property: profileName.

Attribute name	Data type	Default value	Description
queryCloseImplicit	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryCloseImplicit. Values are: 1 (QUERY_CLOSE_IMPLICIT_YES) or 2 (QUERY_CLOSE_IMPLICIT_NO).</p> <p>2 QUERY_CLOSE_IMPLICIT_NO 1 QUERY_CLOSE_IMPLICIT_YES</p>
queryDataSize	<p>int</p> <p>Minimum: 4096</p> <p>Maximum: 65535</p>		JDBC driver property: queryDataSize.
queryTimeoutInterruptProcessingMode	<p>int</p> <ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: queryTimeoutInterruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODE 1 INTERRUPT_PROCESSING_MODE</p>
readOnly	boolean		JDBC driver property: readOnly.
recordTemporalHistory	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: recordTemporalHistory.</p> <p>2 NO 1 YES</p>
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.

Attribute name	Data type	Default value	Description
retryIntervalForClientReroute	A period of time with second precision		JDBC driver property: retryIntervalForClientReroute. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12 		<p>JDBC driver property: securityMechanism. Values are:</p> <ul style="list-style-type: none"> 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY), 11 (KERBEROS_SECURITY), 12 (ENCRYPTED_USER_AND_DATA_SECURITY), 13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY), 15 (PLUGIN_SECURITY), 16 (ENCRYPTED_USER_ONLY_SECURITY), 18 (TLS_CLIENT_CERTIFICATE_SECURITY). <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 18 TLS_CLIENT_CERTIFICATE_SECURITY 15 PLUGIN_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY 16 ENCRYPTED_USER_ONLY_SECURITY 13 ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY 11 KERBEROS_SECURITY 12 ENCRYPTED_USER_AND_DATA_SECURITY</p>
sendDataAsIs	boolean		JDBC driver property: sendDataAsIs.
serverName	string	localhost	Server where the database is running.
sessionTimeZone	string		JDBC driver property: sessionTimeZone.
sqljCloseStmtsWithOpenResultSet	boolean		JDBC driver property: sqljCloseStmtsWithOpenResultSet.

Attribute name	Data type	Default value	Description
sqljEnableClassLoaderSpecificProfiles	boolean		JDBC driver property: sqljEnableClassLoaderSpecificProfiles.
sslConnection	boolean		JDBC driver property: sslConnection.
streamBufferSize	int		JDBC driver property: streamBufferSize.
stripTrailingZerosForDecimalNumbers	Numbers <ul style="list-style-type: none"> • 1 		JDBC driver property: stripTrailingZerosForDecimalNumbers. 2 NO 1 YES
sysSchema	string		JDBC driver property: sysSchema.
timerLevelForQueryTimeOut	<ul style="list-style-type: none"> • 2 • 1 • -1 		JDBC driver property: timerLevelForQueryTimeOut. 2 QUERYTIMEOUT_CONNECTION 1 QUERYTIMEOUT_STATEMENT_L -1 QUERYTIMEOUT_DISABLED
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceFileCount	int		JDBC driver property: traceFileCount.
traceFileSize	int		JDBC driver property: traceFileSize.
traceLevel	int	0	Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
traceOption	<ul style="list-style-type: none"> • 1 • 0 		JDBC driver property: traceOption 1 1 0 0

Attribute name	Data type	Default value	Description
translateForBitData	<ul style="list-style-type: none"> 2 1 		JDBC driver property: translateForBitData. 2 SERVER_ENCODING_REPRESENTATION 1 HEX_REPRESENTATION
updateCountForBatch	<ul style="list-style-type: none"> 2 1 		JDBC driver property: updateCountForBatch. 2 TOTAL_UPDATE_COUNT 1 NO_UPDATE_COUNT
useCachedCursor	boolean		JDBC driver property: useCachedCursor.
useIdentityValLocalForAutoGeneratedKeys	boolean		JDBC driver property: useIdentityValLocalForAutoGeneratedKeys.
useJDBC41DefinitionForGetColumns	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC41DefinitionForGetColumns. 2 NO 1 YES
useJDBC4ColumnNameAndLabelSemantics	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
useTransactionRedirect	boolean		JDBC driver property: useTransactionRedirect.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaNetworkOptimization	boolean		JDBC driver property: xaNetworkOptimization.

oauthProvider > databaseStore > dataSource > properties.derby.client

Data source properties for Derby Network Client JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1527	Port on which to obtain database connections.
retrieveMessageText	boolean	true	JDBC driver property: retrieveMessageText.
securityMechanism	<ul style="list-style-type: none"> 3 7 4 9 8 	3	<p>JDBC driver property: securityMechanism. Values are:</p> <p>3 (CLEAR_TEXT_PASSWORD_SECURITY),</p> <p>4 (USER_ONLY_SECURITY),</p> <p>7 (ENCRYPTED_PASSWORD_SECURITY),</p> <p>8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY),</p> <p>9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY</p> <p>7 ENCRYPTED_PASSWORD_SECURITY</p> <p>4 USER_ONLY_SECURITY</p> <p>9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p> <p>8 STRONG_PASSWORD_SUBSTITUTE_SECURITY</p>

Attribute name	Data type	Default value	Description
serverName	string	localhost	Server where the database is running.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		JDBC driver property: shutdownDatabase. false Do not shut down the database. shutdown Shut down the database when a connection is attempted.
ssl	<ul style="list-style-type: none"> basic off peerAuthentication 		JDBC driver property: ssl. basic basic off off peerAuthentication peerAuthentication
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_XA_CALLS=2048, TRACE_ALL=-1.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.derby.embedded

Data source properties for Derby Embedded JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		<p>JDBC driver property: shutdownDatabase.</p> <p>false Do not shut down the database.</p> <p>shutdown Shut down the database when a connection is attempted.</p>
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.informix

Data source properties for the Informix JDBC driver.

false

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
ifxCLIENT_LOCALE	string		JDBC driver property: ifxCLIENT_LOCALE.
ifxCPMAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMInitPoolSize	int		JDBC driver property: ifxCPMInitPoolSize.
ifxCPMMaxConnections	int		JDBC driver property: ifxCPMMaxConnections.
ifxCPMMaxPoolSize	int		JDBC driver property: ifxCPMMaxPoolSize.
ifxCPMMinAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMMinAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMMinPoolSize	int		JDBC driver property: ifxCPMMinPoolSize.
ifxCPMServiceInterval	A period of time with millisecond precision		JDBC driver property: ifxCPMServiceInterval. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxDBANSIWARN	boolean		JDBC driver property: ifxDBANSIWARN.
ifxDBCENTURY	string		JDBC driver property: ifxDBCENTURY.
ifxDBDATE	string		JDBC driver property: ifxDBDATE.
ifxDBSPACETEMP	string		JDBC driver property: ifxDBSPACETEMP.

Attribute name	Data type	Default value	Description
ifxDBTEMP	string		JDBC driver property: ifxDBTEMP.
ifxDBTIME	string		JDBC driver property: ifxDBTIME.
ifxDBUPSPACE	string		JDBC driver property: ifxDBUPSPACE.
ifxDB_LOCALE	string		JDBC driver property: ifxDB_LOCALE.
ifxDELIMIDENT	boolean		JDBC driver property: ifxDELIMIDENT.
ifxENABLE_TYPE_CACHE	boolean		JDBC driver property: ifxENABLE_TYPE_CACHE.
ifxFET_BUF_SIZE	int		JDBC driver property: ifxFET_BUF_SIZE.
ifxGL_DATE	string		JDBC driver property: ifxGL_DATE.
ifxGL_DATETIME	string		JDBC driver property: ifxGL_DATETIME.
ifxIFXHOST	string	localhost	JDBC driver property: ifxIFXHOST.
ifxIFX_AUTOFREE	boolean		JDBC driver property: ifxIFX_AUTOFREE.
ifxIFX_DIRECTIVES	string		JDBC driver property: ifxIFX_DIRECTIVES.
ifxIFX_LOCK_MODE_WAIT	A period of time with second precision	2s	JDBC driver property: ifxIFX_LOCK_MODE_WAIT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxIFX_SOC_TIMEOUT	A period of time with millisecond precision		JDBC driver property: ifxIFX_SOC_TIMEOUT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxIFX_USEPUT	boolean		JDBC driver property: ifxIFX_USEPUT.

Attribute name	Data type	Default value	Description
ifxIFX_USE_STRENC	boolean		JDBC driver property: ifxIFX_USE_STRENC.
ifxIFX_XASPEC	string	y	JDBC driver property: ifxIFX_XASPEC.
ifxINFORMIXCONRETRY	int		JDBC driver property: ifxINFORMIXCONRETRY.
ifxINFORMIXCONTIME	A period of time with second precision		JDBC driver property: ifxINFORMIXCONTIME. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxINFORMIXOPCACHE	string		JDBC driver property: ifxINFORMIXOPCACHE.
ifxINFORMIXSTACKSIZE	int		JDBC driver property: ifxINFORMIXSTACKSIZE.
ifxJDBCTEMP	string		JDBC driver property: ifxJDBCTEMP.
ifxLDAP_IFXBASE	string		JDBC driver property: ifxLDAP_IFXBASE.
ifxLDAP_PASSWD	string		JDBC driver property: ifxLDAP_PASSWD.
ifxLDAP_URL	string		JDBC driver property: ifxLDAP_URL.
ifxLDAP_USER	string		JDBC driver property: ifxLDAP_USER.
ifxLOBCACHE	int		JDBC driver property: ifxLOBCACHE.
ifxNEWCODESET	string		JDBC driver property: ifxNEWCODESET.
ifxNEWLOCALE	string		JDBC driver property: ifxNEWLOCALE.
ifxNODEFDAC	string		JDBC driver property: ifxNODEFDAC.
ifxOPTCOMPIND	string		JDBC driver property: ifxOPTCOMPIND.
ifxOPTOFC	string		JDBC driver property: ifxOPTOFC.
ifxOPT_GOAL	string		JDBC driver property: ifxOPT_GOAL.
ifxPATH	string		JDBC driver property: ifxPATH.
ifxPDQPRIORITY	string		JDBC driver property: ifxPDQPRIORITY.

Attribute name	Data type	Default value	Description
ifxPLCONFIG	string		JDBC driver property: ifxPLCONFIG.
ifxPLOAD_LO_PATH	string		JDBC driver property: ifxPLOAD_LO_PATH.
ifxPROTOCOLTRACE	int		JDBC driver property: ifxPROTOCOLTRACE.
ifxPROTOCOLTRACEFILE	string		JDBC driver property: ifxPROTOCOLTRACEFILE.
ifxPROXY	string		JDBC driver property: ifxPROXY.
ifxPSORT_DBTEMP	string		JDBC driver property: ifxPSORT_DBTEMP.
ifxPSORT_NPROCS	boolean		JDBC driver property: ifxPSORT_NPROCS.
ifxSECURITY	string		JDBC driver property: ifxSECURITY.
ifxSQLH_FILE	string		JDBC driver property: ifxSQLH_FILE.
ifxSQLH_LOC	string		JDBC driver property: ifxSQLH_LOC.
ifxSQLH_TYPE	string		JDBC driver property: ifxSQLH_TYPE.
ifxSSLCONNECTION	string		JDBC driver property: ifxSSLCONNECTION.
ifxSTMT_CACHE	string		JDBC driver property: ifxSTMT_CACHE.
ifxTRACE	int		JDBC driver property: ifxTRACE.
ifxTRACEFILE	string		JDBC driver property: ifxTRACEFILE.
ifxTRUSTED_CONTEXT	string		JDBC driver property: ifxTRUSTED_CONTEXT.
ifxUSEV5SERVER	boolean		JDBC driver property: ifxUSEV5SERVER.
ifxUSE_DTENV	boolean		JDBC driver property: ifxUSE_DTENV.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
roleName	string		JDBC driver property: roleName.
serverName	string		Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.informix.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for Informix.
false

Attribute name	Data type	Default value	Description
DBANSIWARN	boolean		JDBC driver property: DBANSIWARN.
DBDATE	string		JDBC driver property: DBDATE.
DBPATH	string		JDBC driver property: DBPATH.
DBSPACETEMP	string		JDBC driver property: DBSPACETEMP.
DBTEMP	string		JDBC driver property: DBTEMP.
DBUPSPACE	string		JDBC driver property: DBUPSPACE.
DELIMIDENT	boolean		JDBC driver property: DELIMIDENT.
IFX_DIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_DIRECTIVES. ON ON OFF OFF
IFX_EXTDIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_EXTDIRECTIVES. ON ON OFF OFF
IFX_UPDESC	string		JDBC driver property: IFX_UPDESC.

Attribute name	Data type	Default value	Description
IFX_XASTDCOMPLIANCE_XAEND	<ul style="list-style-type: none"> 0 		JDBC driver property: IFX_XASTDCOMPLIANCE_XAEND. 1 1 0 0
INFORMIXOPCACHE	string		JDBC driver property: INFORMIXOPCACHE.
INFORMIXSTACKSIZE	string		JDBC driver property: INFORMIXSTACKSIZE.
NODEFDAC	<ul style="list-style-type: none"> yes no 		JDBC driver property: NODEFDAC. yes yes no no
OPTCOMPIND	<ul style="list-style-type: none"> 2 1 0 		JDBC driver property: OPTCOMPIND. 2 2 1 1 0 0
OPTOFC	<ul style="list-style-type: none"> 1 0 		JDBC driver property: OPTOFC. 1 1 0 0
PDQPRIORITY	<ul style="list-style-type: none"> HIGH LOW OFF 		JDBC driver property: PDQPRIORITY. HIGH HIGH LOW LOW OFF OFF
PSORT_DBTEMP	string		JDBC driver property: PSORT_DBTEMP.
PSORT_NPROCS	string Maximum: 10		JDBC driver property: PSORT_NPROCS.
STMT_CACHE	<ul style="list-style-type: none"> 1 0 		JDBC driver property: STMT_CACHE. 1 1 0 0
currentLockTimeout	A period of time with second precision	2s	JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean		JDBC driver property: deferPrepares.
driverType	int	4	JDBC driver property: driverType.
enableNamedParameterMarkers	int		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO).
enableSeamlessFailover	int		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO).
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.
keepDynamic	int		JDBC driver property: keepDynamic.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
progressiveStreaming	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: progressiveStreaming. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
queryDataSize	int Minimum: 4096 Maximum: 10485760		JDBC driver property: queryDataSize.

Attribute name	Data type	Default value	Description
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	<p>JDBC driver property: retrieveMessagesFromServerOnGetMessage.</p>
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 9 		<p>JDBC driver property: securityMechanism. Values are: 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY).</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p>
serverName	string	localhost	Server where the database is running.
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.

Attribute name	Data type	Default value	Description
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
useJDBC4ColumnNameAndLabelSemantics	boolean		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO).
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.microsoft.sqlserver

Data source properties for Microsoft SQL Server JDBC Driver.

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database. Example: jdbc:sqlserver://localhost:1433;databaseName=myDB.
applicationIntent	<ul style="list-style-type: none"> ReadOnly ReadWrite 		JDBC driver property: applicationIntent. ReadOnly ReadOnly ReadWrite ReadWrite
applicationName	string		JDBC driver property: applicationName.
authenticationScheme	<ul style="list-style-type: none"> NativeAuthentication JavaKerberos 		JDBC driver property: authenticationScheme. NativeAuthentication NativeAuthentication JavaKerberos JavaKerberos

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
encrypt	boolean		JDBC driver property: encrypt.
failoverPartner	string		JDBC driver property: failoverPartner.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
instanceName	string		JDBC driver property: instanceName.
integratedSecurity	boolean		JDBC driver property: integratedSecurity.
lastUpdateCount	boolean		JDBC driver property: lastUpdateCount.
lockTimeout	A period of time with millisecond precision		JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
multiSubnetFailover	boolean		JDBC driver property: multiSubnetFailover.
packetSize	int Minimum: 512 Maximum: 32767		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.

Attribute name	Data type	Default value	Description
responseBuffering	<ul style="list-style-type: none"> full adaptive 		JDBC driver property: responseBuffering. full full adaptive adaptive
selectMethod	<ul style="list-style-type: none"> direct cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
sendStringParametersAsUnicode	boolean	false	JDBC driver property: sendStringParametersAsUnicode.
sendTimeAsDatetime	boolean		JDBC driver property: sendTimeAsDatetime.
serverName	string	localhost	Server where the database is running.
trustServerCertificate	boolean		JDBC driver property: trustServerCertificate.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
workstationID	string		JDBC driver property: workstationID.
xopenStates	boolean		JDBC driver property: xopenStates.

oauthProvider > databaseStore > dataSource > properties.oracle

Data source properties for Oracle JDBC driver.

false

Attribute name	Data type	Default value	Description
ONSConfiguration	string		JDBC driver property: ONSConfiguration.
TNSEntryName	string		JDBC driver property: TNSEntryName.
URL	string		URL for connecting to the database. Examples: jdbc:oracle:thin:@//localhost:1521/sample or jdbc:oracle:oci:@//localhost:1521/sample.
connectionProperties	string		JDBC driver property: connectionProperties.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
driverType	<ul style="list-style-type: none"> oci thin 	thin	JDBC driver property: driverType. oci oci thin thin
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	string		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1521	Port on which to obtain database connections.
serverName	string	localhost	Server where the database is running.
serviceName	string		JDBC driver property: serviceName.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

oauthProvider > databaseStore > dataSource > properties.sybase

Data source properties for Sybase JDBC driver.

false

Attribute name	Data type	Default value	Description
SERVER_INITIATED_TRANSACTIONS	<ul style="list-style-type: none"> false true 	false	JDBC driver property: SERVER_INITIATED_TRANSACTIONS. false false true true
connectionProperties	string	SELECT_OPENS_CURSOR=true	JDBC driver property: connectionProperties.
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	<ul style="list-style-type: none"> • SSL • socket 		JDBC driver property: networkProtocol. SSL SSL socket socket
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	5000	Port on which to obtain database connections.
resourceManagerName	string		JDBC driver property: resourceManagerName.
serverName	string	localhost	Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
version	int		JDBC driver property: version.

oauthProvider > databaseStore > dataSource > recoveryAuthData

Authentication data for transaction recovery.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

oauthProvider > grantType

An access token grant type (as detailed in the OAuth specification) that is allowed for the provider. The equivalent provider parameter in the full application server profile is `oauth20.grant.types.allowed`.

false

string

oauthProvider > jwtGrantType

The `grant_type` for JWT Token handler

false

Attribute name	Data type	Default value	Description
clockSkew	A period of time with second precision	300s	The time difference allowed between OpenID Connect Client and OpenID Connect Provider systems when they are not synchronized. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
iatRequired	boolean	false	The iat claim in a jwt token is required.
maxJtiCacheSize	long Minimum: 1	10000	The maximum size of cache, which keeps jti data of jwt token, to prevent the jti from being reused.
tokenMaxLifetime	A period of time with second precision	7200s	The time indicates the maximum lifetime of an alive jwt token since its issued-at-time. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

oauthProvider > library

Reference to shared library containing the mediator plugin class.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

oauthProvider > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

oauthProvider > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

oauthProvider > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

oauthProvider > localStore

Clients are defined in server.xml and tokens are cached in the server.

false

Attribute name	Data type	Default value	Description
tokenStoreSize	long	2000	Token store size

oauthProvider > localStore > client

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
applicationType	<ul style="list-style-type: none"> • native • web 	web	The type of application best describing the client. native native web web
displayname	string		Display name of the client.
enabled	boolean	true	Client is enabled if true, disabled if false.

Attribute name	Data type	Default value	Description
functionalUserId	string		A user identifier to be associated with access tokens obtained by this client using the client credentials grant type. When this client parameter is specified, the value is returned in the functional_user_id response parameter from the introspect endpoint.
id	string		A unique configuration ID.
introspectTokens	boolean	false	Boolean value specifying whether the client is allowed to access the introspection endpoint to introspect tokens issued by the authorization server.
name	string		Name of the client (sometimes referred to as the Id).
preAuthorizedScope	string		Space separated list of scope values that the client can use when requesting access tokens that are deemed to have been pre-approved by the resource owner and therefore does not require the resource owner's consent.
scope	string		Specify by spaces the list of scopes of the client.
secret	Reversably encoded password (string)		Secret key of the client.
sessionManaged	boolean	false	Boolean indicating whether the client participates in OpenID session management.
subjectType	<ul style="list-style-type: none"> public 		Subject type requested for response to this client. public public
tokenEndpointAuthMethod	<ul style="list-style-type: none"> client_secret_post none client_secret_basic 	client_secret_basic	The requested authentication method for the token endpoint of the client. client_secret_post client_secret_post none none client_secret_basic client_secret_basic

oauthProvider > localStore > client > functionalUserGroupIds

A list of group ids to be associated with access tokens obtained by this client using the client credentials grant type. When this client parameter is specified, the value is returned in the functional_user_groupIds response parameter from the introspect endpoint.

false

string

oauthProvider > localStore > client > grantTypes

Grant types the client may use.

false

oauthProvider > localStore > client > postLogoutRedirectUris

Array of URLs supplied by the RP to which it may request that the end-user's user agent be redirected using the post_logout_redirect_uri parameter after a logout has been performed.

false

string

oauthProvider > localStore > client > redirect

Array of redirect URIs for use in redirect-based flows such as the authorization code and implicit grant types of the client. The first redirect URI is used as a default, when none is specified in a request.

false

string

oauthProvider > localStore > client > responseTypes

Response types the client may use.

false

oauthProvider > mediatorClassName

Mediator plugin class name. The equivalent provider parameter in the full application server profile is oauth20.mediator.classnames.

false

string

scopeToClaimMap

Specify the claims for the scope.

false

Attribute name	Data type	Default value	Description
address	string	address	Specify a comma-separated list of claims associated with the address scope.
email	string	email, email_verified	Specify a comma-separated list of claims associated with the email scope.
phone	string	phone_number, phone_number_verified	Specify a comma-separated list of claims associated with the phone scope.

Attribute name	Data type	Default value	Description
profile	string	name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, updated_at	Specify a comma-separated list of claims associated with the profile scope.

scopeToClaimMap > property

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Specify the name of the property
value	string		Specify the value of the property

OSGi Application (osgiApplication)

Defines the properties of an osgi application.

- application-bnd
 - security-role
 - group
 - run-as
 - special-subject
 - user

Attribute name	Data type	Default value	Description
autoStart	boolean	true	Indicates whether or not the server automatically starts the application.
id	string		A unique configuration ID.
location	A file, directory or url.		Location of an application expressed as an absolute path or a path relative to the server-level apps directory.
name	string		Name of an application.
suppressUncoveredHttpMethodWarning	boolean	false	Option to suppress uncovered HTTP method warning message during application deployment.

application-bnd

Binds general deployment information included in the application to specific resources.

false

Attribute name	Data type	Default value	Description
version	string		Version of the application bindings specification.

application-bnd > security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a security role.

application-bnd > security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		Group access ID
id	string		A unique configuration ID.
name	string		Name of a group possessing a security role.

application-bnd > security-role > run-as

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of a user required to access a bean from another bean. The value can be stored in clear text or encoded form. To encode the password, use the securityUtility tool with the encode option.
userid	string		ID of a user required to access a bean from another bean.

application-bnd > security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
type	<ul style="list-style-type: none"> • EVERYONE • ALL_AUTHENTICATED_USERS 		One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE. EVERYONE Everyone ALL_AUTHENTICATED_USERS All authenticated users

application-bnd > security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user possessing a security role.

OSGi Applications (osgiApplications)

Settings for all OSGi applications

Attribute name	Data type	Default value	Description
enable.debug	boolean	false	When this option is set to true, OSGi applications that fail to start remain installed to allow the application to be debugged.

OSGi Library (osgiLibrary)

Enables OSGi applications to use packages provided by a shared library.

- library
 - file
 - fileset
 - folder
- package

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
libraryRef	A reference to top level library element (string).		The shared library reference to use for the OSGi library.

library

The shared library reference to use for the OSGi library.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

package

A package export specification for a package that the system makes available for use by OSGi applications.

false

string

Web Server Plugin (pluginConfiguration)

Properties used when generating the web server plugin configuration file

- httpEndpoint
 - accessLogging
 - httpOptions
 - sslOptions
 - tcpOptions

Attribute name	Data type	Default value	Description
connectTimeout	A period of time with second precision	5s	Identifies the maximum amount of time that the application server should maintain a connection with the web server. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
extendedHandshake	boolean	false	If true, the web server plugin uses an extended handshake to determine if the application server is running.
httpEndpointRef	A reference to top level httpEndpoint element (string).	defaultHttpEndpoint	Specify the identifier of the http endpoint to include in the generated plugin-cfg.xml file. The endpoint defines the server in the cluster. The default value is 'defaultHttpEndpoint'.
ipv6Preferred	boolean	false	Used when resolving an application server host name of {null} or {0}, to prefer the type of address when possible
logDirLocation	Path to a directory		Deprecated: Identifies the directory where the http_plugin.log file is located. See Log file name.
logFileName	Path to a file	\${pluginInstallRoot}/logs/\${webserverName}/http_plugin.log	A fully qualified path to to the web server plug-in log file. Directory component must already exist. For Apache-based web servers, a path that begins with a pipe character is interpreted as an external piped logger. If specified, the path overrides logDirLocation.
pluginInstallRoot	string	/opt/IBM/WebSphere/Plugins	Web server plugin installation location in file system of web server host

Attribute name	Data type	Default value	Description
serverIOTimeout	A period of time with second precision	900s	Identifies the maximum amount of time that the web server plugin waits to send a request or receive a response from the application server. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
serverIOTimeoutRetry	int Minimum: -1 Maximum: 65535	-1	Limits the number of request retries after a read or write timeout. The default value, {-1}, applies no additional limits, so retries are limited by the number of available servers in the cluster. A {0} value indicates there should be no retries. This value is scoped to the server cluster and does not apply to connection failures or timeouts due to the HTTP plug-in Connection timeout, or to web socket timeouts.
sslCertlabel	string		Specifies the label of the certificate within the keyring that the plug-in is to use when the web container requests a client certificate from the plug-in.
sslKeyringLocation	string	\${pluginInstallRoot}/ config/\${webserverName}/ plugin-key.kdb	The fully qualified path to the SSL keyring file on the web server host
sslStashfileLocation	string	\${pluginInstallRoot}/ config/\${webserverName}/ plugin-key.sth	The fully qualified path to the SSL stashfile on the web server host

Attribute name	Data type	Default value	Description
waitForContinue	boolean	false	If false (the default value), the web server plugin sends the "Expect: 100-continue" header with HTTP requests that have a message body. When set to true, the web server plugin sends the "Expect: 100-continue" header with every HTTP request. Consider setting this value to true if you have a firewall between the web server and the application server, and are sensitive to requests retries with no request body.
webserviceName	string	webservice1	Name of the web server where this configuration will be used. Used to generate the plugin log file location if that is not specified explicitly by Log file name or directory.
webservicePort	int Minimum: -1 Maximum: 65535	80	Web server HTTP port
webserviceSecurePort	int Minimum: -1 Maximum: 65535	443	Web server HTTPS port
wsServerIOTimeout	A period of time with second precision		Identifies the maximum amount of time that the web server plugin waits to send a request or receive a websocket response from the application server. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
wsServerIdleTimeout	A period of time with second precision		Identifies the maximum amount of time that the web server plugin waits to terminate an idle websocket connection. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

httpEndpoint

Specify the identifier of the http endpoint to include in the generated plugin-cfg.xml file. The endpoint defines the server in the cluster. The default value is 'defaultHttpEndpoint'.

false

Attribute name	Data type	Default value	Description
accessLoggingRef	A reference to top level httpAccessLogging element (string).		HTTP access logging configuration for the endpoint.
enabled	boolean	true	Toggle the availability of an endpoint. When true, this endpoint will be activated by the dispatcher to handle HTTP requests.
host	string	localhost	IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource. Use '*' for all available network interfaces.
httpOptionsRef	A reference to top level httpOptions element (string).	defaultHttpOptions	HTTP protocol options for the endpoint.
httpPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests. Use -1 to disable this port.
httpsPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests secured with SSL (https). Use -1 to disable this port.

Attribute name	Data type	Default value	Description
onError	<ul style="list-style-type: none"> IGNORE FAIL WARN 	WARN	<p>Action to take after a failure to start an endpoint.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>
sslOptionsRef	A reference to top level sslOptions element (string).		SSL protocol options for the endpoint.
tcpOptionsRef	A reference to top level tcpOptions element (string).	defaultTCPOptions	TCP protocol options for the endpoint.

httpEndpoint > accessLogging

HTTP access logging configuration for the endpoint.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	Enable access logging.
filePath	Path to a file	\${server.output.dir}/logs/http_access.log	Directory path and name of the access log file. Standard variable substitutions, such as \${server.output.dir}, can be used when specifying the directory path.
logFormat	string	%h %u %{t}W "%r" %s %b	Specifies the log format that is used when logging client access information.
maxFileSize	int Minimum: 0	20	Maximum size of a log file, in megabytes, before being rolled over; a value of 0 means no limit.
maxFiles	int Minimum: 0	2	Maximum number of log files that will be kept, before the oldest file is removed; a value of 0 means no limit.

httpEndpoint > httpOptions

HTTP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
keepAliveEnabled	boolean	true	Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.
maxKeepAliveRequests	int Minimum: -1	100	Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.
persistTimeout	A period of time with second precision	30s	Amount of time that a socket will be allowed to remain idle between requests. This setting only applies if persistent connections are enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
readTimeout	A period of time with second precision	60s	Amount of time to wait for a read request to complete on a socket after the first read occurs. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
removeServerHeader	boolean	false	Removes server implementation information from HTTP headers and also disables the default Liberty profile welcome page.

Attribute name	Data type	Default value	Description
writeTimeout	A period of time with second precision	60s	Amount of time to wait on a socket for each portion of the response data to be transmitted. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

httpEndpoint > sslOptions

SSL protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
sessionTimeout	A period of time with second precision	1d	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
sslRef	A reference to top level ssl element (string).		The default SSL configuration repertoire. The default value is defaultSSLSettings.
suppressHandshakeErrors	boolean	false	Disable logging of SSL handshake errors. SSL handshake errors can occur during normal operation, however these messages can be useful when SSL is behaving unexpectedly.

httpEndpoint > tcpOptions

TCP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
inactivityTimeout	A period of time with millisecond precision	60s	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
soReuseAddr	boolean	true	Enables immediate rebind to a port with no active listener.

Quick Start Security (quickStartSecurity)

Simple administrative security configuration.

Attribute name	Data type	Default value	Description
userName	string		Single user defined as part of the quick start security configuration. This user is granted the Administrator role.
userPassword	Reversably encoded password (string)		Password for the single user defined as part of the quick start security configuration. It is recommended that you encode this password. To do so, use the securityUtility tool with the encode option.

Remote File Access (remoteFileAccess)

This element contains artifacts that control the level of file access exposed for remote connections.

- readDir
- writeDir

readDir

A directory that remote clients are allowed to read from. There can be multiple readDir elements, and each represents a single directory that may refer to variables or absolute paths. Default is `${wlp.install.dir}`, `${wlp.user.dir}` and `${server.output.dir}`

false

Path to a directory

writeDir

A directory that remote clients are allowed to read from and write to. There can be multiple writeDir elements, and each represents a single directory that may refer to variables or absolute paths. Default is an empty set of directories.

false

Path to a directory

Request Timing (requestTiming)

Provides warnings and diagnostic information for the slow or hung requests.

Attribute name	Data type	Default value	Description
hungRequestThreshold	A period of time with millisecond precision	10m	Duration of time that a request can run before being considered hung. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
includeContextInfo	boolean	true	Indicates if the context information details are included in the log output.
sampleRate	int Minimum: 1	1	Rate at which the sampling should happen for the slow request tracking. To sample one out of every n requests, set sampleRate to n. To sample all requests, set sampleRate to 1.
slowRequestThreshold	A period of time with millisecond precision	10s	Duration of time that a request can run before being considered slow. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

SAML Web SSO 2.0 Authentication (samlWebSso20)

Controls the operation of the Security Assertion Markup Language Web SSO 2.0 Mechanism.

- audiences
- authFilter

- host
- remoteAddress
- requestUrl
- userAgent
- webApp
- authnContextClassRef
- headerName
- pkixTrustEngine
 - crl
 - trustedIssuers
 - x509Certificate

Attribute name	Data type	Default value	Description
allowCreate	boolean		Allow the IdP to create a new account if the requesting user does not have one.
allowCustomCacheKey	boolean	true	Allow generating a custom cache key to access the authentication cache and get the subject.
authFilterRef	A reference to top level authFilter element (string).		Specifies the authentication filter reference.

Attribute name	Data type	Default value	Description
authnContextComparisonType	<ul style="list-style-type: none"> • minimum • better • maximum • exact 	exact	<p>When an authnContextClassRef is specified, the authnContextComparisonType can be set.</p> <p>minimum Minimum. The authentication context in the authentication statement must be at least as strong as one of the authentication contexts specified.</p> <p>better Better. The authentication context in the authentication statement must be stronger than any one of the authentication contexts specified.</p> <p>maximum Maximum. The authentication context in the authentication statement must be as strong as possible without exceeding the strength of at least one of the authentication contexts specified.</p> <p>exact Exact. The authentication context in the authentication statement must be an exact match of at least one of the authentication contexts specified.</p>

Attribute name	Data type	Default value	Description
authnRequestTime	A period of time with millisecond precision	10m	Specifies the life time period of an authnRequest which is generated and sent from the service provider to an IdP for requesting a SAML Token. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
authnRequestsSigned	boolean	true	Indicates whether the <samlp:AuthnRequest> messages sent by this service provider will be signed.
clockSkew	A period of time with millisecond precision	5m	This is used to specify the allowed clock skew in minutes when validating the SAML token. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
createSession	boolean	true	Specifies whether to create an HttpSession if the current HttpSession does not exist.
customizeNameIDFormat	string		Specifies the customized URI reference corresponding to a name identifier format that is not defined in the SAML core specification.
disableLtpaCookie	boolean	true	Do not create an LTPA Token during processing of the SAML Assertion. Create a cookie of the specific Service Provider instead.
enabled	boolean	true	The service provider is enabled if true and disabled if false.

Attribute name	Data type	Default value	Description
errorPageURL	string		Specifies an error page to be displayed if the SAML validation fails. If this attribute is not specified, and the received SAML is invalid, the user will be redirected back to the SAML IdP to restart SSO.
forceAuthn	boolean	false	Indicates whether the IdP should force the user to re-authenticate.
groupIdIdentifier	string		Specifies a SAML attribute that is used as the name of the group that the authenticated principal is a member of. There is no default value.
httpsRequired	boolean	true	Enforce using SSL communication when accessing a SAML WebSSO service provider end point such as acs or metadata.
id	string		A unique configuration ID.
idpMetadata	string	`\${server.config.dir}/resources/security/idpMetadata.xml`	Specifies the IdP metadata file.
inboundPropagation	<ul style="list-style-type: none"> • none • required 	none	<p>Controls the operation of the Security Assertion Markup Language Web SSO 2.0 for the inbound propagation of the Web Services Mechanisms.</p> <p>none %inboundPropagation.none required %inboundPropagation.required</p>
includeTokenInSubject	boolean	true	Specifies whether to include a SAML assertion in the subject.
includeX509InSPMetadata	boolean	true	Specifies whether to include the x509 certificate in the Liberty SP metadata.
isPassive	boolean	false	Indicates IdP must not take control of the end user interface.
keyAlias	string		Key alias name to locate the private key for signing and decryption. This is optional if the keystore has exactly one key entry, or if it has one key with an alias of 'samlsp'.

Attribute name	Data type	Default value	Description
keyStoreRef	A reference to top level keyStore element (string).		A keystore containing the private key for the signing of the AuthnRequest, and the decryption of EncryptedAssertion element. The default is the server's default.
loginPageURL	string		Specifies the SAML IdP login application URL to which an unauthenticated request is redirected. This attribute triggers IdP-initiated SSO, and it is only required for IdP-initiated SSO.
mapToUserRegistry	<ul style="list-style-type: none"> • User • No • Group 	No	<p>Specifies how to map an identity to a registry user. The options are No, User, and Group. The default is No, and the user registry is not used to create the user subject.</p> <p>User Map a SAML identity to a user defined in the registry</p> <p>No Do not map a SAML identity to a user or group in the registry</p> <p>Group Map a SAML identity to a group defined in the user registry</p>

Attribute name	Data type	Default value	Description
nameIDFormat	<ul style="list-style-type: none"> • encrypted • customize • persistent • x509SubjectName • email • transient • entity • unspecified • kerberos • windowsDomainQualifiedName 	email	<p>Specifies the URI reference corresponding to a name identifier format defined in the SAML core specification.</p> <p>encrypted urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted</p> <p>customize Customized Name ID Format.</p> <p>persistent urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</p> <p>x509SubjectName urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</p> <p>email urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</p> <p>transient urn:oasis:names:tc:SAML:2.0:nameid-format:transient</p> <p>entity urn:oasis:names:tc:SAML:2.0:nameid-format:entity</p> <p>unspecified urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</p> <p>kerberos urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos</p> <p>windowsDomainQualifiedName urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName</p>
reAuthnCushion	A period of time with millisecond precision	0m	<p>The time period to authenticate again when a SAML Assertion is about to expire, which is indicated by either the statement NotOnOrAfter or the attribute SessionNotOnOrAfter of the SAML Assertion. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.</p>

Attribute name	Data type	Default value	Description
reAuthnOnAssertionExpire	boolean	false	Authenticate the incoming HTTP request again when a SAML Assertion is about to expire.
realmIdentifier	string		Specifies a SAML attribute that is used as the realm name. If no value is specified, the Issuer SAML assertion element value is used.
realmName	string		Specifies a realm name when mapToUserRegistry is set to No or Group.
sessionNotOnOrAfter	A period of time with millisecond precision	120m	Indicates an upper bound on SAML session durations, after which the Liberty SP should ask the user to re-authenticate to the IdP. If the SAML token returned from the IdP does not contain a sessionNotOnOrAfter assertion, the value specified by this attribute is used. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
signatureMethodAlgorithm	<ul style="list-style-type: none"> • SHA256 • SHA128 • SHA1 	SHA256	<p>Indicates the required algorithm by this service provider.</p> <p>SHA256 SHA-256 signature algorithm</p> <p>SHA128 %signatureMethodAlgorithm.SHA128</p> <p>SHA1 SHA-1 signature algorithm</p>
spHostAndPort	string		Specifies a SAML service provider host name and port number.
targetPageUrl	string		The default landing page for the IdP-initiated SSO if the relayState is missing.

Attribute name	Data type	Default value	Description
tokenReplayTimeout	A period of time with millisecond precision	30m	This property is used to specify how long the Liberty SP should prevent token replay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
userIdentifier	string		Specifies a SAML attribute that is used as the user principal name in the subject. If no value is specified, the NameID SAML assertion element value is used.
userUniqueIdentifier	string		Specifies a SAML attribute that is used as the unique user name as it applies to the WSCredential in the subject. The default is the same as the userIdentifier attribute value.
wantAssertionsSigned	boolean	true	Indicates a requirement for the <saml:Assertion> elements received by this service provider to be signed.

audiences

The list of audiences which are trusted to verify the audience of the SAML Token. If the value is "ANY", then all audiences are trusted.

false

string

authFilter

Specifies the authentication filter reference.

false

authFilter > host

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

authFilter > remoteAddress

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ip	string		Specifies the IP address.
matchType	<ul style="list-style-type: none"> • lessThan • equals • greaterThan • contains • notContain 	contains	Specifies the match type. lessThan Less than equals Equals greaterThan Greater than contains Contains notContain Not contain

authFilter > requestUrl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
urlPattern	string		Specifies the URL pattern.

authFilter > userAgent

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
agent	string		Specifies the user agent
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain

authFilter > webApp

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

authnContextClassRef

A URI reference identifying an authentication context class that describes the authentication context declaration. The default is null.

false

string

headerName

The header name of the HTTP request which stores the SAML Token.

false

string

pkixTrustEngine

Specifies the PKIX trust information that is used to evaluate the trustworthiness and validity of XML signatures in a SAML response. Do not specify multiple pkixTrustEngine in a samlWebSso20.

false

Attribute name	Data type	Default value	Description
trustAnchorRef	A reference to top level keyStore element (string).		A keystore containing the public key necessary for verifying the signature of the SAMLResponse and Assertion.

pkixTrustEngine > crl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
path	string		Specifies the path to the crl.

pkixTrustEngine > trustedIssuers

Specifies the identities of trusted IdP issuers. If the value is "ALL_ISSUERS", then all IdP identities are trusted.

false

string

pkixTrustEngine > x509Certificate

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
path	string		Specifies the path to the x509 certificate.

IBM SecureWay Directory Server LDAP Filters (securewayLdapFilterProperties)

Specifies the list of default IBM SecureWay Directory Server LDAP filters.

Attribute name	Data type	Default value	Description
groupFilter	string	(&(cn=%v)(!(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))	An LDAP filter clause for searching the user registry for groups.
groupIdMap	string	*:cn	An LDAP filter that maps the name of a group to an LDAP entry.
groupMemberIdMap	string	groupOfNames:member;groupOfUniqueNames:uniqueMember	An LDAP filter that identifies user to group memberships.
id	string		A unique configuration ID.
userFilter	string	(&(uid=%v)(objectclass=ePerson))	An LDAP filter clause for searching the user registry for users.
userIdMap	string	*:uid	An LDAP filter that maps the name of a user to an LDAP entry.

Spnego Authentication (spnego)

Controls the operation of the Simple and Protected GSS-API Negotiation Mechanism.

- authFilter
 - host
 - remoteAddress
 - requestUrl

- userAgent
- webApp

Attribute name	Data type	Default value	Description
authFilterRef	A reference to top level authFilter element (string).		Specifies the authentication filter reference.
canonicalHostName	boolean	true	Controls whether you want to use the canonical host name.
disableFailOverToAppAuthType	boolean	true	Specifies that SPNEGO is used to log in to WebSphere Application Server first. However, if the login fails, then the application authentication mechanism is used to log in to the WebSphere Application Server.
includeClientGSSCredentialsInSubject	boolean	true	Specifies whether the client delegation credentials should be stored in a client subject.
krb5Config	string		Specifies the fully qualified Kerberos configuration path and name. Standard variable substitutions, such as \${server.config.dir}, can be used when specifying the directory path.
krb5Keytab	string		Specifies the fully qualified Kerberos keytab path and name. Standard variable substitutions, such as \${server.config.dir}, can be used when specifying the directory path. The Kerberos keytab file contains a list of keys that are analogous to user passwords. It is important for hosts to protect their Kerberos keytab files by storing them on the local disk.
ntlmTokenReceivedErrorPageURL	string		Specifies the URL of a resource that contains the content which SPNEGO includes in the HTTP response, which is displayed by the browser client application.
servicePrincipalNames	string		Specifies a list of Kerberos service principal names separated by a comma.

Attribute name	Data type	Default value	Description
spnegoNotSupportedErrorPageURL	string		Specifies the URL of a resource that contains the content which SPNEGO includes in the HTTP response that is displayed by the browser client application if it does not support SPNEGO authentication.
trimKerberosRealmNameFromPrincipal	boolean	true	Specifies whether SPNEGO removes the suffix of the Kerberos principal user name, starting from the @ that precedes the Kerberos realm name. If this attribute is set to true, the suffix of the principal user name is removed. If this attribute is set to false, the suffix of the principal name is retained.

authFilter

Specifies the authentication filter reference.

false

authFilter > host

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

authFilter > remoteAddress

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
ip	string		Specifies the IP address.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> lessThan equals greaterThan contains notContain 	contains	Specifies the match type. lessThan Less than equals Equals greaterThan Greater than contains Contains notContain Not contain

authFilter > requestUrl

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> equals contains notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
urlPattern	string		Specifies the URL pattern.

authFilter > userAgent

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
agent	string		Specifies the user agent
id	string		A unique configuration ID.
matchType	<ul style="list-style-type: none"> equals contains notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain

authFilter > webApp

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
matchType	<ul style="list-style-type: none"> • equals • contains • notContain 	contains	Specifies the match type. equals Equals contains Contains notContain Not contain
name	string		Specifies the name.

SSL Repertoire (ssl)

An SSL repertoire with an ID, a defined keystore, and an optional truststore.

Attribute name	Data type	Default value	Description
clientAuthentication	boolean	false	Specifies whether client authentication is enabled. If set to true then client authentication is required and the client must provide a certificate for the server trusts.
clientAuthenticationSupported	boolean	false	Specifies whether a client authentication is supported. If set to true then the client authentication support means the server will check trust from a client if the client presents a certificate.
clientKeyAlias	string		Specifies the alias of the certificate in the keystore that is used as the key to send to a server that has client authentication enabled. This attribute is only needed if the keystore has more than one key entry.
enabledCiphers	string		Specifies a custom list of ciphers. Separate each cipher in the list with a space. The supported cipher will depend on the underlying JRE used. Please check the JRE for valid ciphers.
id	string		A unique configuration ID.
keyStoreRef	A reference to top level keyStore element (string).		A keystore containing key entries for the SSL repertoire. This attribute is required.

Attribute name	Data type	Default value	Description
securityLevel	<ul style="list-style-type: none"> • MEDIUM • CUSTOM • HIGH • LOW 	HIGH	<p>Specifies the cipher suite group used by the SSL handshake. HIGH are 3DES and 128 bit and higher ciphers, MEDIUM are DES and 40 bit ciphers, LOW are ciphers without encryption. If the enabledCiphers attribute is used the securityLevel list is ignored.</p> <p>MEDIUM %repertoire.MEDIUM</p> <p>CUSTOM %repertoire.CUSTOM</p> <p>HIGH Cipher suites 3DES and 128 bit and higher</p> <p>LOW %repertoire.LOW</p>
serverKeyAlias	string		Specifies the alias of the certificate in the keystore used as the server's key. This attribute is only needed if the keystore has more than one key entry.
sslProtocol	string		The SSL handshake protocol. Protocol values can be found in the documentation for the underlying JRE's Java Secure Socket Extension (JSSE) provider. When using the IBM JRE the default value is SSL_TLS and when using the Oracle JRE the default value is SSL.
trustStoreRef	A reference to top level keyStore element (string).	\${keyStoreRef}	A keystore containing trusted certificate entries used by the SSL repertoire for signing verification. This attribute is optional. If unspecified, the same keystore is used for both key and trusted certificate entries.

SSL Default Repertoire (sslDefault)

The default repertoire for SSL services.

Attribute name	Data type	Default value	Description
sslRef	A reference to top level ssl element (string).	defaultSSLConfig	The name of the SSL repertoire that will be used as the default. The default SSL repertoire called defaultSSLConfig is used if none is specified.

SSL Options (sslOptions)

The SSL protocol configuration for a transport.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
sessionTimeout	A period of time with second precision	1d	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
sslRef	A reference to top level ssl element (string).		The default SSL configuration repertoire. The default value is defaultSSLSettings.
suppressHandshakeErrors	boolean	false	Disable logging of SSL handshake errors. SSL handshake errors can occur during normal operation, however these messages can be useful when SSL is behaving unexpectedly.

TCP Options (tcpOptions)

Defines TCP protocol settings.

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.

Attribute name	Data type	Default value	Description
inactivityTimeout	A period of time with millisecond precision	60s	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
soReuseAddr	boolean	true	Enables immediate rebind to a port with no active listener.
addressExcludeList	string		A list of addresses that are not allowed to make inbound connections on this channel. IPv4 or IPv6 addresses can be specified. All values in an IPv4 or IPv6 address must be represented by a number, or an asterisk that is used as a wildcard character.
addressIncludeList	string		A list of addresses that are allowed to make inbound connections on this channel. IPv4 or IPv6 addresses can be specified. All values in an IPv4 or IPv6 address must be represented by a number, or an asterisk that is used as a wildcard character.
hostNameExcludeList	string		A list of host names that are not allowed to make inbound connections on this channel. Host names can start with an asterisk, which is used as a wildcard character. However, an asterisk must not appear elsewhere in the address. For example, *.abc.com is valid, but *.abc.* is invalid.

Attribute name	Data type	Default value	Description
hostNameIncludeList	string		A list of host names that are allowed to make inbound connections on this channel. Host names can start with an asterisk, which is used as a wildcard character. However, an asterisk must not appear elsewhere in the address. For example, *.abc.com is valid, but *.abc.* is invalid.
maxOpenConnections	integer	128000	The maximum number of open connections per HTTP Endpoint. The range of possible values is between 1 and 1280000. This value is not strictly enforced to the exact value that is configured due to the timing of connections opening and closing.

For more information, see TCP transport channel settings.

Timed Operation (timedOperation)

Timed operations help WebSphere Application Server administrators see when certain actions in their application server are operating more slowly than expected.

Attribute name	Data type	Default value	Description
enableReport	boolean	true	Enables periodic generation of report to the logs detailing the ten longest timed operations, grouped by type, and sorted within each group by expected duration
maxNumberTimedOperations	int	10000	A warning is logged when the total number of timed operations reaches this value.
reportFrequency	A period of time with hour precision		Frequency of generating report to the logs detailing the ten longest timed operations, grouped by type, and sorted within each group by average of actual duration. Specify a positive integer followed by the unit of time, which can be hours (h). For example, specify 12 hours as 12h.

Transaction Manager (transaction)

Configuration properties for the Transaction Manager service

- dataSource
 - connectionManager
 - containerAuthData
 - jaasLoginContextEntry
 - jdbcDriver
 - library
 - file
 - fileset
 - folder
 - properties
 - properties.datadirect.sqlserver
 - properties.db2.i.native
 - properties.db2.i.toolbox
 - properties.db2.jcc
 - properties.derby.client
 - properties.derby.embedded
 - properties.informix
 - properties.informix.jcc
 - properties.microsoft.sqlserver
 - properties.oracle
 - properties.sybase
 - recoveryAuthData

Attribute name	Data type	Default value	Description
acceptHeuristicHazard	boolean	true	Specifies whether all applications on this server accept the possibility of a heuristic hazard occurring in a two-phase transaction that contains a one-phase resource.
clientInactivityTimeout	A period of time with second precision	60s	Maximum duration between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
dataSourceRef	A reference to top level dataSource element (string).		This is an optional property. By default the transaction service stores its recovery logs in a file. As an alternative it is possible to store the logs in an RDBMS. This is achieved by setting this property which defines a non-transactional data source where the transaction logs will be stored.
defaultMaxShutdownDelay	A period of time with second precision	2s	Default maximum shutdown delay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
enableLoggingForHeuristicReporting	boolean	false	Specifies whether the application server logs about-to-commit-one-phase-resource events from transactions that involve both a one-phase commit resource and two-phase commit resources.
heuristicRetryInterval	A period of time with second precision	60s	Amount of time that the application server waits before retrying a completion signal, such as commit or rollback, after a transient exception from a resource manager or remote partner. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
heuristicRetryWait	int	5	The number of times that the application server retries a completion signal, such as commit or rollback. Retries occur after a transient exception from a resource manager or remote partner.

Attribute name	Data type	Default value	Description
lpsHeuristicCompletion	<ul style="list-style-type: none"> • COMMIT • MANUAL • ROLLBACK 	ROLLBACK	<p>Specifies the direction that is used to complete a transaction that has a heuristic outcome; either the application server commits or rolls back the transaction, or depends on manual completion by the administrator. Allowed values are: COMMIT, ROLLBACK and MANUAL</p> <p>COMMIT Commit</p> <p>MANUAL Manual</p> <p>ROLLBACK Rollback</p>
propogatedOrBMTTranLifetime	Period of time with second precision	0	Upper limit of the transaction timeout for transactions that run in this server. This value should be greater than or equal to the value specified for the total transaction timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
recoverOnStartup	boolean	false	Specifies whether the server should begin transaction recovery at server startup.
recoveryGroup	string		Name of the recovery group that this server belongs too. Members of a recovery group can recover the transaction logs of other servers in the group.
recoveryIdentity	string		Unique identity of this server for transaction peer recovery.
timeoutGracePeriodEnabled	boolean	false	Specifies whether there is a delay between a transaction timeout and the abnormal ending of the servant region that was running the transaction.

Attribute name	Data type	Default value	Description
totalTranLifetimeTimeout	A period of time with second precision	120s	Default maximum time allowed for transactions started on this server to complete. Any such transactions that do not complete before this timeout occurs are rolled back. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
transactionLogDBTableSuffix	string		When recovery logs are stored in an RDBMS table, this property allows the table name to be post-pended with a string to make it unique for this Server.
transactionLogDirectory	string	\${server.output.dir}/tranlog/	A directory for this server where the transaction service stores log files for recovery.
transactionLogSize	int	1024	Specifies the size of transaction log files in Kilobytes.
waitForRecovery	boolean	false	Specifies whether the server should wait for transaction recovery to complete before accepting new transactional work.

dataSource

This is an optional property. By default the transaction service stores its recovery logs in a file. As an alternative it is possible to store the logs in an RDBMS. This is achieved by setting this property which defines a non-transactional data source where the transaction logs will be stored.

false

Attribute name	Data type	Default value	Description
beginTranForResultSetScrolling	boolean	true	Attempt transaction enlistment when result set scrolling interfaces are used.
beginTranForVendorAPIs	boolean	true	Attempt transaction enlistment when vendor interfaces are used.

Attribute name	Data type	Default value	Description
commitOrRollbackOnCleanup	<ul style="list-style-type: none"> commit rollback 		<p>Determines how to clean up connections that might be in a database unit of work (AutoCommit=false) when the connection is closed or returned to the pool.</p> <p>commit Clean up the connection by committing.</p> <p>rollback Clean up the connection by rolling back.</p>
connectionManagerRef	A reference to top level connectionManager element (string).		Connection manager for a data source.
connectionSharing	<ul style="list-style-type: none"> MatchOriginalRequest MatchCurrentState 	MatchOriginalRequest	<p>Specifies how connections are matched for sharing.</p> <p>MatchOriginalRequest When sharing connections, match based on the original connection request.</p> <p>MatchCurrentState When sharing connections, match based on the current state of the connection.</p>
containerAuthDataRef	A reference to top level authData element (string).		Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

Attribute name	Data type	Default value	Description
enableConnectionCasting	boolean	false	Indicates that connections obtained from the data source should be castable to interface classes that the JDBC vendor connection implementation implements. Enabling this option incurs additional overhead on each getConnection operation. If vendor JDBC interfaces are needed less frequently, it might be more efficient to leave this option disabled and use Connection.unwrap(interface) only where it is needed.
isolationLevel	<ul style="list-style-type: none"> • TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT 		<p>Default transaction isolation level.</p> <p>TRANSACTION_REPEATABLE_READ Dirty reads and non-repeatable reads are prevented; phantom reads can occur.</p> <p>TRANSACTION_READ_COMMITTED Dirty reads are prevented; non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SERIALIZABLE Dirty reads, non-repeatable reads and phantom reads are prevented.</p> <p>TRANSACTION_READ_UNCOMMITTED Dirty reads, non-repeatable reads and phantom reads can occur.</p> <p>TRANSACTION_SNAPSHOT Snapshot isolation for Microsoft SQL Server JDBC Driver and DataDirect Connect for JDBC driver.</p>

Attribute name	Data type	Default value	Description
jaasLoginContextEntryRef	A reference to top level jaasLoginContextEntry element (string).		JAAS login context entry for authentication.
jdbcDriverRef	A reference to top level jdbcDriver element (string).		JDBC driver for a data source.
jndiName	string		JNDI name for a data source.
queryTimeout	A period of time with second precision		Default query timeout for SQL statements. In a JTA transaction, syncQueryTimeoutWithTransactionTimeout can override this default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
recoveryAuthDataRef	A reference to top level authData element (string).		Authentication data for transaction recovery.
statementCacheSize	int Minimum: 0	10	Maximum number of cached statements per connection.
supplementalJDBCTrace	boolean		Supplements the JDBC driver trace that is logged when JDBC driver trace is enabled in bootstrap.properties. JDBC driver trace specifications include: com.ibm.ws.database.logwriter, com.ibm.ws.db2.logwriter, com.ibm.ws.derby.logwriter, com.ibm.ws.informix.logwriter, com.ibm.ws.oracle.logwriter, com.ibm.ws.sqlserver.logwriter, com.ibm.ws.sybase.logwriter
syncQueryTimeoutWithTransactionTimeout	boolean	false	Use the time remaining (if any) in a JTA transaction as the default query timeout for SQL statements.
transactional	boolean	true	Enable participation in transactions that are managed by the application server.

Attribute name	Data type	Default value	Description
type	<ul style="list-style-type: none"> javax.sql.DataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource 		Type of data source. javax.sql.DataSource javax.sql.DataSource javax.sql.XADataSource javax.sql.XADataSource javax.sql.ConnectionPoolDataSource javax.sql.ConnectionPoolDataSource

dataSource > connectionManager

Connection manager for a data source.

false

Attribute name	Data type	Default value	Description
agedTimeout	A period of time with second precision	-1	Amount of time before a physical connection can be discarded by pool maintenance. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
connectionTimeout	A period of time with second precision	30s	Amount of time after which a connection request times out. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxConnectionsPerThread	int Minimum: 0		Limits the number of open connections on each thread.

Attribute name	Data type	Default value	Description
maxIdleTime	A period of time with second precision	30m	Amount of time after which an unused or idle connection can be discarded during pool maintenance, if doing so does not reduce the pool below the minimum size. A value of -1 disables this timeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maxPoolSize	int Minimum: 0	50	Maximum number of physical connections for a pool. A value of 0 means unlimited.
minPoolSize	int Minimum: 0		Minimum number of physical connections to maintain in the pool. The pool is not pre-populated. Aged timeout can override the minimum.
numConnectionsPerThreadLocal	int Minimum: 0		Caches the specified number of connections for each thread.

Attribute name	Data type	Default value	Description
purgePolicy	<ul style="list-style-type: none"> • ValidateAllConnections • FailingConnectionOnly • EntirePool 	EntirePool	<p>Specifies which connections to destroy when a stale connection is detected in a pool.</p> <p>ValidateAllConnections When a stale connection is detected, connections are tested and those found to be bad are closed.</p> <p>FailingConnectionOnly When a stale connection is detected, only the connection which was found to be bad is closed.</p> <p>EntirePool When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed.</p>
reapTime	A period of time with second precision	3m	Amount of time between runs of the pool maintenance thread. A value of -1 disables pool maintenance. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

dataSource > containerAuthData

Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

dataSource > jaasLoginContextEntry

JAAS login context entry for authentication.

false

Attribute name	Data type	Default value	Description
loginModuleRef	List of references to top level jaasLoginModule elements (comma-separated string).	hashtable,userNameAndPassword,certificateURL	Word certificateURL of a JAAS login module.
name	string		Name of a JAAS configuration entry.

dataSource > jdbcDriver

JDBC driver for a data source.

false

Attribute name	Data type	Default value	Description
javax.sql.ConnectionPoolDataSource	string		JDBC driver implementation of javax.sql.ConnectionPoolDataSource.
javax.sql.DataSource	string		JDBC driver implementation of javax.sql.DataSource.
javax.sql.XADataSource	string		JDBC driver implementation of javax.sql.XADataSource.
libraryRef	A reference to top level library element (string).		Identifies JDBC driver JARs and native files.

dataSource > jdbcDriver > library

Identifies JDBC driver JARs and native files.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

dataSource > jdbcDriver > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

dataSource > jdbcDriver > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

dataSource > jdbcDriver > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

dataSource > properties

List of JDBC vendor properties for the data source. For example, `databaseName="dbname" serverName="localhost" portNumber="50000"`.

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database.
databaseName	string		JDBC driver property: databaseName.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
serverName	string		Server where the database is running.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.datadirect.sqlserver

Data source properties for the DataDirect Connect for JDBC driver for Microsoft SQL Server.

false

Attribute name	Data type	Default value	Description
JDBCBehavior	<ul style="list-style-type: none"> • 1 • 0 	0	JDBC driver property: JDBCBehavior. Values are: 0 (JDBC 4.0) or 1 (JDBC 3.0). 1 JDBC 3.0 0 JDBC 4.0
XATransactionGroup	string		JDBC driver property: XATransactionGroup.
XMLDescribeType	<ul style="list-style-type: none"> • longvarbinary • longvarchar 		JDBC driver property: XMLDescribeType. longvarbinary longvarbinary longvarchar longvarchar
accountingInfo	string		JDBC driver property: accountingInfo.
alternateServers	string		JDBC driver property: alternateServers.
alwaysReportTriggerResults	boolean		JDBC driver property: alwaysReportTriggerResults.
applicationName	string		JDBC driver property: applicationName.
authenticationMethod	<ul style="list-style-type: none"> • ntlm • userIdPassword • kerberos • auto 		JDBC driver property: authenticationMethod. ntlm ntlm userIdPassword userIdPassword kerberos kerberos auto auto
bulkLoadBatchSize	long		JDBC driver property: bulkLoadBatchSize.
bulkLoadOptions	long		JDBC driver property: bulkLoadOptions.
clientHostName	string		JDBC driver property: clientHostName.
clientUser	string		JDBC driver property: clientUser.

Attribute name	Data type	Default value	Description
codePageOverride	string		JDBC driver property: codePageOverride.
connectionRetryCount	int		JDBC driver property: connectionRetryCount.
connectionRetryDelay	A period of time with second precision		JDBC driver property: connectionRetryDelay. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
convertNull	int		JDBC driver property: convertNull.
databaseName	string		JDBC driver property: databaseName.
dateTimeInputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeInputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
dateTimeOutputParameterType	<ul style="list-style-type: none"> • dateTime • dateTimeOffset • auto 		JDBC driver property: dateTimeOutputParameterType. dateTime dateTime dateTimeOffset dateTimeOffset auto auto
describeInputParameters	<ul style="list-style-type: none"> • describeIfString • noDescribe • describeIfDateTime • describeAll 		JDBC driver property: describeInputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll

Attribute name	Data type	Default value	Description
describeOutputParameters	<ul style="list-style-type: none"> describeIfString noDescribe describeIfDateTime describeAll 		JDBC driver property: describeOutputParameters. describeIfString describeIfString noDescribe noDescribe describeIfDateTime describeIfDateTime describeAll describeAll
enableBulkLoad	boolean		JDBC driver property: enableBulkLoad.
enableCancelTimeout	boolean		JDBC driver property: enableCancelTimeout.
encryptionMethod	<ul style="list-style-type: none"> loginSSL requestSSL SSL noEncryption 		JDBC driver property: encryptionMethod. loginSSL loginSSL requestSSL requestSSL SSL SSL noEncryption noEncryption
failoverGranularity	<ul style="list-style-type: none"> disableIntegrityCheck atomicWithRepositioning nonAtomic atomic 		JDBC driver property: failoverGranularity. disableIntegrityCheck disableIntegrityCheck atomicWithRepositioning atomicWithRepositioning nonAtomic nonAtomic atomic atomic
failoverMode	<ul style="list-style-type: none"> connect select extended 		JDBC driver property: failoverMode. connect connect select select extended extended
failoverPreconnect	boolean		JDBC driver property: failoverPreconnect.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
initializationString	string		JDBC driver property: initializationString.
insensitiveResultSetBufferSize	int		JDBC driver property: insensitiveResultSetBufferSize.

Attribute name	Data type	Default value	Description
javaDoubleToString	boolean		JDBC driver property: javaDoubleToString.
loadBalancing	boolean		JDBC driver property: loadBalancing.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
longDataCacheSize	int Minimum: -1		JDBC driver property: longDataCacheSize.
netAddress	string		JDBC driver property: netAddress.
packetSize	int Minimum: -1 Maximum: 128		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
queryTimeout	A period of time with second precision		JDBC driver property: queryTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
resultsetMetaDataOptions	int		JDBC driver property: resultSetMetaDataOptions.
selectMethod	<ul style="list-style-type: none"> • direct • cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
serverName	string	localhost	Server where the database is running.
snapshotSerializable	boolean		JDBC driver property: snapshotSerializable.

Attribute name	Data type	Default value	Description
spyAttributes	string		JDBC driver property: spyAttributes.
stringInputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringInputParameterType. varchar varchar nvarchar nvarchar
stringOutputParameterType	<ul style="list-style-type: none"> varchar nvarchar 	varchar	JDBC driver property: stringOutputParameterType. varchar varchar nvarchar nvarchar
suppressConnectionWarnings	boolean		JDBC driver property: suppressConnectionWarnings.
transactionMode	<ul style="list-style-type: none"> explicit implicit 		JDBC driver property: transactionMode. explicit explicit implicit implicit
truncateFractionalSeconds	boolean		JDBC driver property: truncateFractionalSeconds.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
useServerSideUpdatableCursors	boolean		JDBC driver property: useServerSideUpdatableCursors.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
validateServerCertificate	boolean		JDBC driver property: validateServerCertificate.

dataSource > properties.db2.i.native

Data source properties for the IBM DB2 for i Native JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> read only all read call 	all	JDBC driver property: access. read only read only all all read call read call

Attribute name	Data type	Default value	Description
autoCommit	boolean	true	JDBC driver property: autoCommit.
batchStyle	<ul style="list-style-type: none"> • 2.1 • 2.0 	2.0	JDBC driver property: batchStyle. 2.1 2.1 2.0 2.0
behaviorOverride	int		JDBC driver property: behaviorOverride.
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive
dataTruncation	string	true	JDBC driver property: dataTruncation.
databaseName	string	*LOCAL	JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy
dateSeparator	<ul style="list-style-type: none"> • \, • b • . • / • - 		JDBC driver property: dateSeparator. \, The comma character (,). b The character b . The period character (.). / The forward slash character (/). - The dash character (-).
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		JDBC driver property: decimalSeparator. \, The comma character (,). . The period character (.).
directMap	boolean	true	JDBC driver property: directMap.
doEscapeProcessing	boolean	true	JDBC driver property: doEscapeProcessing.
fullErrors	boolean		JDBC driver property: fullErrors.
libraries	string		JDBC driver property: libraries.
lobThreshold	int Maximum: 500000	0	JDBC driver property: lobThreshold.

Attribute name	Data type	Default value	Description
lockTimeout	A period of time with second precision	0	JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 63
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
networkProtocol	int		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.
prefetch	boolean	true	JDBC driver property: prefetch.
queryOptimizeGoal	<ul style="list-style-type: none"> • 2 • 1 	2	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO). 2 *ALLIO 1 *FIRSTIO

Attribute name	Data type	Default value	Description
reuseObjects	boolean	true	JDBC driver property: reuseObjects.
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
systemNaming	boolean	false	JDBC driver property: systemNaming.
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • \, • b • : • . 		JDBC driver property: timeSeparator. \, The comma character (,). b The character b : The colon character (:). . The period character (.).
trace	boolean		JDBC driver property: trace.
transactionTimeout	A period of time with second precision	0	JDBC driver property: transactionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateHex	<ul style="list-style-type: none"> • binary • character 	character	JDBC driver property: translateHex. binary binary character character
useBlockInsert	boolean	false	JDBC driver property: useBlockInsert.

Attribute name	Data type	Default value	Description
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.db2.i.toolbox

Data source properties for the IBM DB2 for i Toolbox JDBC driver.

false

Attribute name	Data type	Default value	Description
access	<ul style="list-style-type: none"> • read only • all • read call 	all	JDBC driver property: access. read only read only all all read call read call
behaviorOverride	int		JDBC driver property: behaviorOverride.
bidiImplicitReordering	boolean	true	JDBC driver property: bidiImplicitReordering.
bidiNumericOrdering	boolean	false	JDBC driver property: bidiNumericOrdering.
bidiStringType	int		JDBC driver property: bidiStringType.
bigDecimal	boolean	true	JDBC driver property: bigDecimal.
blockCriteria	<ul style="list-style-type: none"> • 2 • 1 • 0 	2	JDBC driver property: blockCriteria. Values are: 0 (no record blocking), 1 (block if FOR FETCH ONLY is specified), 2 (block if FOR UPDATE is specified). 2 2 1 1 0 0

Attribute name	Data type	Default value	Description
blockSize	<ul style="list-style-type: none"> • 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256 	32	JDBC driver property: blockSize. 512 512 128 128 0 0 32 32 64 64 16 16 8 8 256 256
cursorHold	boolean	false	JDBC driver property: cursorHold.
cursorSensitivity	<ul style="list-style-type: none"> • asensitive • sensitive • insensitive 	asensitive	JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). asensitive asensitive sensitive sensitive insensitive insensitive
dataCompression	boolean	true	JDBC driver property: dataCompression.
dataTruncation	boolean	true	JDBC driver property: dataTruncation.
databaseName	string		JDBC driver property: databaseName.
dateFormat	<ul style="list-style-type: none"> • dmy • iso • eur • ymd • julian • jis • usa • mdy 		JDBC driver property: dateFormat. dmy dmy iso iso eur eur ymd ymd julian julian jis jis usa usa mdy mdy

Attribute name	Data type	Default value	Description
dateSeparator	<ul style="list-style-type: none"> • • \, • . • / • - 		<p>JDBC driver property: dateSeparator.</p> <p>The space character ().</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p> <p>/ The forward slash character (/).</p> <p>- The dash character (-).</p>
decimalSeparator	<ul style="list-style-type: none"> • \, • . 		<p>JDBC driver property: decimalSeparator.</p> <p>\, The comma character (,).</p> <p>. The period character (.).</p>
driver	<ul style="list-style-type: none"> • toolbox • native 	toolbox	<p>JDBC driver property: driver.</p> <p>toolbox toolbox</p> <p>native native</p>
errors	<ul style="list-style-type: none"> • full • basic 	basic	<p>JDBC driver property: errors.</p> <p>full full</p> <p>basic basic</p>
extendedDynamic	boolean	false	JDBC driver property: extendedDynamic.
extendedMetaData	boolean	false	JDBC driver property: extendedMetaData.
fullOpen	boolean	false	JDBC driver property: fullOpen.
holdInputLocators	boolean	true	JDBC driver property: holdInputLocators.
holdStatements	boolean	false	JDBC driver property: holdStatements.
isolationLevelSwitchingSupport	boolean	false	JDBC driver property: isolationLevelSwitchingSupport.
keepAlive	boolean		JDBC driver property: keepAlive.
lazyClose	boolean	false	JDBC driver property: lazyClose.
libraries	string		JDBC driver property: libraries.

Attribute name	Data type	Default value	Description
lobThreshold	int Minimum: 0 Maximum: 16777216	0	JDBC driver property: lobThreshold.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
maximumPrecision	<ul style="list-style-type: none"> • 31 • 63 	31	JDBC driver property: maximumPrecision. 31 31 63 64
maximumScale	int Minimum: 0 Maximum: 63	31	JDBC driver property: maximumScale.
metaDataSource	int Minimum: 0 Maximum: 1	1	JDBC driver property: metaDataSource.
minimumDivideScale	int Minimum: 0 Maximum: 9	0	JDBC driver property: minimumDivideScale.
naming	<ul style="list-style-type: none"> • system • sql 	sql	JDBC driver property: naming. system system sql sql
package	string		JDBC driver property: package.
packageAdd	boolean	true	JDBC driver property: packageAdd.
packageCCSID	<ul style="list-style-type: none"> • 13488 • 1200 	13488	JDBC driver property: packageCCSID. Values are: 1200 (UCS-2) or 13488 (UTF-16). 13488 13488 (UTF-16) 1200 1200 (UCS-2)
packageCache	boolean	false	JDBC driver property: packageCache.

Attribute name	Data type	Default value	Description
packageCriteria	<ul style="list-style-type: none"> default select 	default	JDBC driver property: packageCriteria. default default select select
packageError	<ul style="list-style-type: none"> exception none warning 	warning	JDBC driver property: packageError. exception exception none none warning warning
packageLibrary	string	QGPL	JDBC driver property: packageLibrary.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
prefetch	boolean	true	JDBC driver property: prefetch.
prompt	boolean	false	JDBC driver property: prompt.
proxyServer	string		JDBC driver property: proxyServer.
qaqqiniLibrary	string		JDBC driver property: qaqqiniLibrary.
queryOptimizeGoal	int Minimum: 0 Maximum: 2	0	JDBC driver property: queryOptimizeGoal. Values are: 1 (*FIRSTIO) or 2 (*ALLIO).
receiveBufferSize	int Minimum: 1		JDBC driver property: receiveBufferSize.
remarks	<ul style="list-style-type: none"> system sql 	system	JDBC driver property: remarks. system system sql sql
rollbackCursorHold	boolean	false	JDBC driver property: rollbackCursorHold.
savePasswordWhenSerialized	boolean	false	JDBC driver property: savePasswordWhenSerialized.
secondaryUrl	string		JDBC driver property: secondaryUrl.
secure	boolean	false	JDBC driver property: secure.
sendBufferSize	int Minimum: 1		JDBC driver property: sendBufferSize.

Attribute name	Data type	Default value	Description
serverName	string		Server where the database is running.
serverTraceCategories	int	0	JDBC driver property: serverTraceCategories.
soLinger	A period of time with second precision		JDBC driver property: soLinger. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
soTimeout	A period of time with millisecond precision		JDBC driver property: soTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
sort	<ul style="list-style-type: none"> • hex • table • language 	hex	JDBC driver property: sort. hex hex table table language language
sortLanguage	string		JDBC driver property: sortLanguage.
sortTable	string		JDBC driver property: sortTable.
sortWeight	<ul style="list-style-type: none"> • unqiue • shared 		JDBC driver property: sortWeight. unqiue unique shared shared
tcpNoDelay	boolean		JDBC driver property: tcpNoDelay.
threadUsed	boolean	true	JDBC driver property: threadUsed.

Attribute name	Data type	Default value	Description
timeFormat	<ul style="list-style-type: none"> • iso • eur • jis • usa • hms 		JDBC driver property: timeFormat. iso iso eur eur jis jis usa usa hms hms
timeSeparator	<ul style="list-style-type: none"> • • \, • : • . 		JDBC driver property: timeSeparator. The space character (). \, The comma character (,). : The colon character (:). . The period character (.).
toolboxTrace	<ul style="list-style-type: none"> • diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning 		JDBC driver property: toolboxTrace. diagnostic diagnostic information information conversion conversion error error thread thread proxy proxy none none datastream datastream pcml pcml all all jdbc jdbc warning warning
trace	boolean		JDBC driver property: trace.
translateBinary	boolean	false	JDBC driver property: translateBinary.
translateBoolean	boolean	true	JDBC driver property: translateBoolean.

Attribute name	Data type	Default value	Description
translateHex	<ul style="list-style-type: none"> binary character 	character	JDBC driver property: translateHex. binary binary character character
trueAutoCommit	boolean	false	JDBC driver property: trueAutoCommit.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaLooselyCoupledSupport	int Minimum: 0 Maximum: 1	0	JDBC driver property: xaLooselyCoupledSupport.

dataSource > properties.db2.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for DB2.

false

Attribute name	Data type	Default value	Description
activateDatabase	int		JDBC driver property: activateDatabase.
alternateGroupDatabaseName	string		JDBC driver property: alternateGroupDatabaseName.
alternateGroupPortNumber	string		JDBC driver property: alternateGroupPortNumber.
alternateGroupServerName	string		JDBC driver property: alternateGroupServerName.
blockingReadConnectionTimeout	A period of time with second precision		JDBC driver property: blockingReadConnectionTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
clientAccountingInformation	string		JDBC driver property: clientAccountingInformation.
clientApplicationInformation	string		JDBC driver property: clientApplicationInformation.
clientRerouteAlternatePortNumber	string		JDBC driver property: clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	string		JDBC driver property: clientRerouteAlternateServerName.

Attribute name	Data type	Default value	Description
clientUser	string		JDBC driver property: clientUser.
clientWorkstation	string		JDBC driver property: clientWorkstation.
connectionCloseWithInFlightTransaction	Transaction • 1		JDBC driver property: connectionCloseWithInFlightTransaction. 2 CONNECTION_CLOSE_WITH_ROLLBACK 1 CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	int		JDBC driver property: currentAlternateGroupEntry.
currentFunctionPath	string		JDBC driver property: currentFunctionPath.
currentLocaleLcCtype	string		JDBC driver property: currentLocaleLcCtype.
currentLockTimeout	A period of time with second precision		JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
currentPackagePath	string		JDBC driver property: currentPackagePath.
currentPackageSet	string		JDBC driver property: currentPackageSet.
currentSQLID	string		JDBC driver property: currentSQLID.
currentSchema	string		JDBC driver property: currentSchema.
cursorSensitivity	• 2 • 1 • 0		JDBC driver property: cursorSensitivity. Values are: 0 (TYPE_SCROLL_SENSITIVE_STATIC), 1 (TYPE_SCROLL_SENSITIVE_DYNAMIC), 2 (TYPE_SCROLL_ASENSITIVE). 2 TYPE_SCROLL_ASENSITIVE 1 TYPE_SCROLL_SENSITIVE_DYNAMIC 0 TYPE_SCROLL_SENSITIVE_STATIC
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean	true	JDBC driver property: deferPrepares.

Attribute name	Data type	Default value	Description
driverType	<ul style="list-style-type: none"> 2 4 	4	JDBC driver property: driverType. 2 Type 2 JDBC driver. 4 Type 4 JDBC driver.
enableAlternateGroupSeamlessACR	boolean		JDBC driver property: enableAlternateGroupSeamlessACR.
enableClientAffinitiesList	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableClientAffinitiesList. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableExtendedDescribe	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedDescribe. 2 NO 1 YES
enableExtendedIndicators	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableExtendedIndicators. 2 NO 1 YES
enableNamedParameterMarkers	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSeamlessFailover	<ul style="list-style-type: none"> 2 1 		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeInputStreams	boolean		JDBC driver property: fullyMaterializeInputStreams.
fullyMaterializeInputStreamsOnBatchExecution	<ul style="list-style-type: none"> 1 		JDBC driver property: fullyMaterializeInputStreamsOnBatchExecution. 2 NO 1 YES
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.

Attribute name	Data type	Default value	Description
implicitRollbackOption	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: implicitRollbackOption.</p> <p>2 IMPLICIT_ROLLBACK_OPTION_N 1 IMPLICIT_ROLLBACK_OPTION_N 0 IMPLICIT_ROLLBACK_OPTION_N</p>
interruptProcessingMode	<ul style="list-style-type: none"> • 2 • 1 • 0 		<p>JDBC driver property: interruptProcessingMode.</p> <p>2 INTERRUPT_PROCESSING_MODI 1 INTERRUPT_PROCESSING_MODI 0 INTERRUPT_PROCESSING_MODI</p>
keepAliveTimeOut	A period of time with second precision		<p>JDBC driver property: keepAliveTimeOut. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
keepDynamic	int		JDBC driver property: keepDynamic.
kerberosServerPrincipal	string		JDBC driver property: kerberosServerPrincipal.
loginTimeout	A period of time with second precision		<p>JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.</p>
maxConnCachedParamBufferSize	int		JDBC driver property: maxConnCachedParamBufferSize.
maxRetriesForClientReroute	int		JDBC driver property: maxRetriesForClientReroute.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	50000	Port on which to obtain database connections.
profileName	string		JDBC driver property: profileName.

Attribute name	Data type	Default value	Description
queryCloseImplicit	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: queryCloseImplicit. Values are: 1 (QUERY_CLOSE_IMPLICIT_YES) or 2 (QUERY_CLOSE_IMPLICIT_NO). 2 QUERY_CLOSE_IMPLICIT_NO 1 QUERY_CLOSE_IMPLICIT_YES
queryDataSize	int Minimum: 4096 Maximum: 65535		JDBC driver property: queryDataSize.
queryTimeoutInterruptProcessingMode	int <ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: queryTimeoutInterruptProcessingMode. 2 INTERRUPT_PROCESSING_MODE_C 1 INTERRUPT_PROCESSING_MODE_S
readOnly	boolean		JDBC driver property: readOnly.
recordTemporalHistory	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: recordTemporalHistory. 2 NO 1 YES
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT). 2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT). 2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.

Attribute name	Data type	Default value	Description
retryIntervalForClientReroute	A period of time with second precision		JDBC driver property: retryIntervalForClientReroute. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12 		<p>JDBC driver property: securityMechanism. Values are:</p> <ul style="list-style-type: none"> 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY), 11 (KERBEROS_SECURITY), 12 (ENCRYPTED_USER_AND_DATA_SECURITY), 13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY), 15 (PLUGIN_SECURITY), 16 (ENCRYPTED_USER_ONLY_SECURITY), 18 (TLS_CLIENT_CERTIFICATE_SECURITY). <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 18 TLS_CLIENT_CERTIFICATE_SECURITY 15 PLUGIN_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY 16 ENCRYPTED_USER_ONLY_SECURITY 13 ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY 11 KERBEROS_SECURITY 12 ENCRYPTED_USER_AND_DATA_SECURITY</p>
sendDataAsIs	boolean		JDBC driver property: sendDataAsIs.
serverName	string	localhost	Server where the database is running.
sessionTimeZone	string		JDBC driver property: sessionTimeZone.
sqljCloseStmtsWithOpenResultSet	boolean		JDBC driver property: sqljCloseStmtsWithOpenResultSet.

Attribute name	Data type	Default value	Description
sqljEnableClassLoaderSpecificProfiles	boolean		JDBC driver property: sqljEnableClassLoaderSpecificProfiles.
sslConnection	boolean		JDBC driver property: sslConnection.
streamBufferSize	int		JDBC driver property: streamBufferSize.
stripTrailingZerosForDecimalNumbers	int • 1		JDBC driver property: stripTrailingZerosForDecimalNumbers. 2 NO 1 YES
sysSchema	string		JDBC driver property: sysSchema.
timerLevelForQueryTimeOut	int • 2 • 1 • -1		JDBC driver property: timerLevelForQueryTimeOut. 2 QUERYTIMEOUT_CONNECTION_LEVEL 1 QUERYTIMEOUT_STATEMENT_LEVEL -1 QUERYTIMEOUT_DISABLED
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceFileCount	int		JDBC driver property: traceFileCount.
traceFileSize	int		JDBC driver property: traceFileSize.
traceLevel	int	0	Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
traceOption	int • 1 • 0		JDBC driver property: traceOption 1 1 0 0

Attribute name	Data type	Default value	Description
translateForBitData	<ul style="list-style-type: none"> 2 1 		JDBC driver property: translateForBitData. 2 SERVER_ENCODING_REPRESENTATION 1 HEX_REPRESENTATION
updateCountForBatch	<ul style="list-style-type: none"> 2 1 		JDBC driver property: updateCountForBatch. 2 TOTAL_UPDATE_COUNT 1 NO_UPDATE_COUNT
useCachedCursor	boolean		JDBC driver property: useCachedCursor.
useIdentityValLocalForAutoGeneratedKeys	boolean		JDBC driver property: useIdentityValLocalForAutoGeneratedKeys.
useJDBC41DefinitionForGetColumns	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC41DefinitionForGetColumns. 2 NO 1 YES
useJDBC4ColumnNameAndLabelSemantics	<ul style="list-style-type: none"> 2 1 		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
useTransactionRedirect	boolean		JDBC driver property: useTransactionRedirect.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
xaNetworkOptimization	boolean		JDBC driver property: xaNetworkOptimization.

dataSource > properties.derby.client

Data source properties for Derby Network Client JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1527	Port on which to obtain database connections.
retrieveMessageText	boolean	true	JDBC driver property: retrieveMessageText.
securityMechanism	<ul style="list-style-type: none"> 3 7 4 9 8 	3	<p>JDBC driver property: securityMechanism. Values are:</p> <p>3 (CLEAR_TEXT_PASSWORD_SECURITY),</p> <p>4 (USER_ONLY_SECURITY),</p> <p>7 (ENCRYPTED_PASSWORD_SECURITY),</p> <p>8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY),</p> <p>9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY</p> <p>7 ENCRYPTED_PASSWORD_SECURITY</p> <p>4 USER_ONLY_SECURITY</p> <p>9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p> <p>8 STRONG_PASSWORD_SUBSTITUTE_SECURITY</p>

Attribute name	Data type	Default value	Description
serverName	string	localhost	Server where the database is running.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		JDBC driver property: shutdownDatabase. false Do not shut down the database. shutdown Shut down the database when a connection is attempted.
ssl	<ul style="list-style-type: none"> basic off peerAuthentication 		JDBC driver property: ssl. basic basic off off peerAuthentication peerAuthentication
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_XA_CALLS=2048, TRACE_ALL=-1.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.derby.embedded

Data source properties for Derby Embedded JDBC driver.

false

Attribute name	Data type	Default value	Description
connectionAttributes	string		JDBC driver property: connectionAttributes.

Attribute name	Data type	Default value	Description
createDatabase	<ul style="list-style-type: none"> false create 		<p>JDBC driver property: createDatabase.</p> <p>false Do not automatically create the database.</p> <p>create When the first connection is established, automatically create the database if it doesn't exist.</p>
databaseName	string		JDBC driver property: databaseName.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
shutdownDatabase	<ul style="list-style-type: none"> false shutdown 		<p>JDBC driver property: shutdownDatabase.</p> <p>false Do not shut down the database.</p> <p>shutdown Shut down the database when a connection is attempted.</p>
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.informix

Data source properties for the Informix JDBC driver.

false

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
ifxCLIENT_LOCALE	string		JDBC driver property: ifxCLIENT_LOCALE.
ifxCPMAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMInitPoolSize	int		JDBC driver property: ifxCPMInitPoolSize.
ifxCPMMaxConnections	int		JDBC driver property: ifxCPMMaxConnections.
ifxCPMMaxPoolSize	int		JDBC driver property: ifxCPMMaxPoolSize.
ifxCPMMinAgeLimit	A period of time with second precision		JDBC driver property: ifxCPMMinAgeLimit. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxCPMMinPoolSize	int		JDBC driver property: ifxCPMMinPoolSize.
ifxCPMServiceInterval	A period of time with millisecond precision		JDBC driver property: ifxCPMServiceInterval. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxDBANSIWARN	boolean		JDBC driver property: ifxDBANSIWARN.
ifxDBCENTURY	string		JDBC driver property: ifxDBCENTURY.
ifxDBDATE	string		JDBC driver property: ifxDBDATE.
ifxDBSPACETEMP	string		JDBC driver property: ifxDBSPACETEMP.

Attribute name	Data type	Default value	Description
ifxDBTEMP	string		JDBC driver property: ifxDBTEMP.
ifxDBTIME	string		JDBC driver property: ifxDBTIME.
ifxDBUPSPACE	string		JDBC driver property: ifxDBUPSPACE.
ifxDB_LOCALE	string		JDBC driver property: ifxDB_LOCALE.
ifxDELIMIDENT	boolean		JDBC driver property: ifxDELIMIDENT.
ifxENABLE_TYPE_CACHE	boolean		JDBC driver property: ifxENABLE_TYPE_CACHE.
ifxFET_BUF_SIZE	int		JDBC driver property: ifxFET_BUF_SIZE.
ifxGL_DATE	string		JDBC driver property: ifxGL_DATE.
ifxGL_DATETIME	string		JDBC driver property: ifxGL_DATETIME.
ifxIFXHOST	string	localhost	JDBC driver property: ifxIFXHOST.
ifxIFX_AUTOFREE	boolean		JDBC driver property: ifxIFX_AUTOFREE.
ifxIFX_DIRECTIVES	string		JDBC driver property: ifxIFX_DIRECTIVES.
ifxIFX_LOCK_MODE_WAIT	A period of time with second precision	2s	JDBC driver property: ifxIFX_LOCK_MODE_WAIT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxIFX_SOC_TIMEOUT	A period of time with millisecond precision		JDBC driver property: ifxIFX_SOC_TIMEOUT. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
ifxIFX_USEPUT	boolean		JDBC driver property: ifxIFX_USEPUT.

Attribute name	Data type	Default value	Description
ifxIFX_USE_STRENC	boolean		JDBC driver property: ifxIFX_USE_STRENC.
ifxIFX_XASPEC	string	y	JDBC driver property: ifxIFX_XASPEC.
ifxINFORMIXCONRETRY	int		JDBC driver property: ifxINFORMIXCONRETRY.
ifxINFORMIXCONTIME	A period of time with second precision		JDBC driver property: ifxINFORMIXCONTIME. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
ifxINFORMIXOPCACHE	string		JDBC driver property: ifxINFORMIXOPCACHE.
ifxINFORMIXSTACKSIZE	int		JDBC driver property: ifxINFORMIXSTACKSIZE.
ifxJDBCTEMP	string		JDBC driver property: ifxJDBCTEMP.
ifxLDAP_IFXBASE	string		JDBC driver property: ifxLDAP_IFXBASE.
ifxLDAP_PASSWD	string		JDBC driver property: ifxLDAP_PASSWD.
ifxLDAP_URL	string		JDBC driver property: ifxLDAP_URL.
ifxLDAP_USER	string		JDBC driver property: ifxLDAP_USER.
ifxLOBCACHE	int		JDBC driver property: ifxLOBCACHE.
ifxNEWCODESET	string		JDBC driver property: ifxNEWCODESET.
ifxNEWLOCALE	string		JDBC driver property: ifxNEWLOCALE.
ifxNODEFDAC	string		JDBC driver property: ifxNODEFDAC.
ifxOPTCOMPIND	string		JDBC driver property: ifxOPTCOMPIND.
ifxOPTOFC	string		JDBC driver property: ifxOPTOFC.
ifxOPT_GOAL	string		JDBC driver property: ifxOPT_GOAL.
ifxPATH	string		JDBC driver property: ifxPATH.
ifxPDQPRIORITY	string		JDBC driver property: ifxPDQPRIORITY.

Attribute name	Data type	Default value	Description
ifxPLCONFIG	string		JDBC driver property: ifxPLCONFIG.
ifxPLOAD_LO_PATH	string		JDBC driver property: ifxPLOAD_LO_PATH.
ifxPROTOCOLTRACE	int		JDBC driver property: ifxPROTOCOLTRACE.
ifxPROTOCOLTRACEFILE	string		JDBC driver property: ifxPROTOCOLTRACEFILE.
ifxPROXY	string		JDBC driver property: ifxPROXY.
ifxPSORT_DBTEMP	string		JDBC driver property: ifxPSORT_DBTEMP.
ifxPSORT_NPROCS	boolean		JDBC driver property: ifxPSORT_NPROCS.
ifxSECURITY	string		JDBC driver property: ifxSECURITY.
ifxSQLH_FILE	string		JDBC driver property: ifxSQLH_FILE.
ifxSQLH_LOC	string		JDBC driver property: ifxSQLH_LOC.
ifxSQLH_TYPE	string		JDBC driver property: ifxSQLH_TYPE.
ifxSSLCONNECTION	string		JDBC driver property: ifxSSLCONNECTION.
ifxSTMT_CACHE	string		JDBC driver property: ifxSTMT_CACHE.
ifxTRACE	int		JDBC driver property: ifxTRACE.
ifxTRACEFILE	string		JDBC driver property: ifxTRACEFILE.
ifxTRUSTED_CONTEXT	string		JDBC driver property: ifxTRUSTED_CONTEXT.
ifxUSEV5SERVER	boolean		JDBC driver property: ifxUSEV5SERVER.
ifxUSE_DTENV	boolean		JDBC driver property: ifxUSE_DTENV.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
roleName	string		JDBC driver property: roleName.
serverName	string		Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.informix.jcc

Data source properties for the IBM Data Server Driver for JDBC and SQLJ for Informix.

false

Attribute name	Data type	Default value	Description
DBANSIWARN	boolean		JDBC driver property: DBANSIWARN.
DBDATE	string		JDBC driver property: DBDATE.
DBPATH	string		JDBC driver property: DBPATH.
DBSPACETEMP	string		JDBC driver property: DBSPACETEMP.
DBTEMP	string		JDBC driver property: DBTEMP.
DBUPSPACE	string		JDBC driver property: DBUPSPACE.
DELIMIDENT	boolean		JDBC driver property: DELIMIDENT.
IFX_DIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_DIRECTIVES. ON ON OFF OFF
IFX_EXTDIRECTIVES	<ul style="list-style-type: none"> • ON • OFF 		JDBC driver property: IFX_EXTDIRECTIVES. ON ON OFF OFF
IFX_UPDESC	string		JDBC driver property: IFX_UPDESC.

Attribute name	Data type	Default value	Description
IFX_XASTDCOMPLIANCE_XAEND	<ul style="list-style-type: none"> 0 		JDBC driver property: IFX_XASTDCOMPLIANCE_XAEND. 1 1 0 0
INFORMIXOPCACHE	string		JDBC driver property: INFORMIXOPCACHE.
INFORMIXSTACKSIZE	string		JDBC driver property: INFORMIXSTACKSIZE.
NODEFDAC	<ul style="list-style-type: none"> yes no 		JDBC driver property: NODEFDAC. yes yes no no
OPTCOMPIND	<ul style="list-style-type: none"> 2 1 0 		JDBC driver property: OPTCOMPIND. 2 2 1 1 0 0
OPTOFC	<ul style="list-style-type: none"> 1 0 		JDBC driver property: OPTOFC. 1 1 0 0
PDQPRIORITY	<ul style="list-style-type: none"> HIGH LOW OFF 		JDBC driver property: PDQPRIORITY. HIGH HIGH LOW LOW OFF OFF
PSORT_DBTEMP	string		JDBC driver property: PSORT_DBTEMP.
PSORT_NPROCS	string Maximum: 10		JDBC driver property: PSORT_NPROCS.
STMT_CACHE	<ul style="list-style-type: none"> 1 0 		JDBC driver property: STMT_CACHE. 1 1 0 0
currentLockTimeout	A period of time with second precision	2s	JDBC driver property: currentLockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
deferPrepares	boolean		JDBC driver property: deferPrepares.
driverType	int	4	JDBC driver property: driverType.
enableNamedParameterMarkers	int		JDBC driver property: enableNamedParameterMarkers. Values are: 1 (YES) or 2 (NO).
enableSeamlessFailover	int		JDBC driver property: enableSeamlessFailover. Values are: 1 (YES) or 2 (NO).
enableSysplexWLB	boolean		JDBC driver property: enableSysplexWLB.
fetchSize	int		JDBC driver property: fetchSize.
fullyMaterializeLobData	boolean		JDBC driver property: fullyMaterializeLobData.
keepDynamic	int		JDBC driver property: keepDynamic.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1526	Port on which to obtain database connections.
progressiveStreaming	<ul style="list-style-type: none"> • 2 • 1 		JDBC driver property: progressiveStreaming. Values are: 1 (YES) or 2 (NO). 2 NO 1 YES
queryDataSize	int Minimum: 4096 Maximum: 10485760		JDBC driver property: queryDataSize.

Attribute name	Data type	Default value	Description
resultSetHoldability	<ul style="list-style-type: none"> • 2 • 1 		<p>JDBC driver property: resultSetHoldability. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
resultSetHoldabilityForCatalogQueries	<ul style="list-style-type: none"> • 1 		<p>JDBC driver property: resultSetHoldabilityForCatalogQueries. Values are: 1 (HOLD_CURSORS_OVER_COMMIT) or 2 (CLOSE_CURSORS_AT_COMMIT).</p> <p>2 CLOSE_CURSORS_AT_COMMIT 1 HOLD_CURSORS_OVER_COMMIT</p>
retrieveMessagesFromServerOnGetMessage	boolean	true	JDBC driver property: retrieveMessagesFromServerOnGetMessage.
securityMechanism	<ul style="list-style-type: none"> • 3 • 7 • 4 • 9 		<p>JDBC driver property: securityMechanism. Values are: 3 (CLEAR_TEXT_PASSWORD_SECURITY), 4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY), 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY).</p> <p>3 CLEAR_TEXT_PASSWORD_SECURITY 7 ENCRYPTED_PASSWORD_SECURITY 4 USER_ONLY_SECURITY 9 ENCRYPTED_USER_AND_PASSWORD_SECURITY</p>
serverName	string	localhost	Server where the database is running.
traceDirectory	string		JDBC driver property: traceDirectory.
traceFile	string		JDBC driver property: traceFile.
traceFileAppend	boolean		JDBC driver property: traceFileAppend.

Attribute name	Data type	Default value	Description
traceLevel	int		Bitwise combination of the following constant values: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTS=32, TRACE_DRDA_FLOWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQLJ=1024, TRACE_META_CALLS=8192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=32768, TRACE_SYSTEM_MONITOR=131072, TRACE_TRACEPOINTS=262144, TRACE_ALL=-1.
useJDBC4ColumnNameAndLabelSemantics	boolean		JDBC driver property: useJDBC4ColumnNameAndLabelSemantics. Values are: 1 (YES) or 2 (NO).
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.microsoft.sqlserver

Data source properties for Microsoft SQL Server JDBC Driver.

false

Attribute name	Data type	Default value	Description
URL	string		URL for connecting to the database. Example: jdbc:sqlserver://localhost:1433;databaseName=myDB.
applicationIntent	<ul style="list-style-type: none"> ReadOnly ReadWrite 		JDBC driver property: applicationIntent. ReadOnly ReadOnly ReadWrite ReadWrite
applicationName	string		JDBC driver property: applicationName.
authenticationScheme	<ul style="list-style-type: none"> NativeAuthentication JavaKerberos 		JDBC driver property: authenticationScheme. NativeAuthentication NativeAuthentication JavaKerberos JavaKerberos

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
encrypt	boolean		JDBC driver property: encrypt.
failoverPartner	string		JDBC driver property: failoverPartner.
hostNameInCertificate	string		JDBC driver property: hostNameInCertificate.
instanceName	string		JDBC driver property: instanceName.
integratedSecurity	boolean		JDBC driver property: integratedSecurity.
lastUpdateCount	boolean		JDBC driver property: lastUpdateCount.
lockTimeout	A period of time with millisecond precision		JDBC driver property: lockTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
multiSubnetFailover	boolean		JDBC driver property: multiSubnetFailover.
packetSize	int Minimum: 512 Maximum: 32767		JDBC driver property: packetSize.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int		Port on which to obtain database connections.

Attribute name	Data type	Default value	Description
responseBuffering	<ul style="list-style-type: none"> full adaptive 		JDBC driver property: responseBuffering. full full adaptive adaptive
selectMethod	<ul style="list-style-type: none"> direct cursor 		JDBC driver property: selectMethod. direct direct cursor cursor
sendStringParametersAsUnicode	boolean	false	JDBC driver property: sendStringParametersAsUnicode.
sendTimeAsDatetime	boolean		JDBC driver property: sendTimeAsDatetime.
serverName	string	localhost	Server where the database is running.
trustServerCertificate	boolean		JDBC driver property: trustServerCertificate.
trustStore	string		JDBC driver property: trustStore.
trustStorePassword	Reversably encoded password (string)		JDBC driver property: trustStorePassword.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
workstationID	string		JDBC driver property: workstationID.
xopenStates	boolean		JDBC driver property: xopenStates.

dataSource > properties.oracle

Data source properties for Oracle JDBC driver.

false

Attribute name	Data type	Default value	Description
ONSConfiguration	string		JDBC driver property: ONSConfiguration.
TNSEntryName	string		JDBC driver property: TNSEntryName.
URL	string		URL for connecting to the database. Examples: jdbc:oracle:thin://localhost:1521/sample or jdbc:oracle:oci://localhost:1521/sample.
connectionProperties	string		JDBC driver property: connectionProperties.

Attribute name	Data type	Default value	Description
databaseName	string		JDBC driver property: databaseName.
driverType	<ul style="list-style-type: none"> oci thin 	thin	JDBC driver property: driverType. oci oci thin thin
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	string		JDBC driver property: networkProtocol.
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	1521	Port on which to obtain database connections.
serverName	string	localhost	Server where the database is running.
serviceName	string		JDBC driver property: serviceName.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.

dataSource > properties.sybase

Data source properties for Sybase JDBC driver.

false

Attribute name	Data type	Default value	Description
SERVER_INITIATED_TRANSACTIONS	<ul style="list-style-type: none"> false true 	false	JDBC driver property: SERVER_INITIATED_TRANSACTIONS. false false true true
connectionProperties	string	SELECT_OPENS_CURSOR=1	JDBC driver property: connectionProperties.
databaseName	string		JDBC driver property: databaseName.

Attribute name	Data type	Default value	Description
loginTimeout	A period of time with second precision		JDBC driver property: loginTimeout. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
networkProtocol	<ul style="list-style-type: none"> • SSL • socket 		JDBC driver property: networkProtocol. SSL SSL socket socket
password	Reversably encoded password (string)		It is recommended to use a container managed authentication alias instead of configuring this property.
portNumber	int	5000	Port on which to obtain database connections.
resourceManagerName	string		JDBC driver property: resourceManagerName.
serverName	string	localhost	Server where the database is running.
user	string		It is recommended to use a container managed authentication alias instead of configuring this property.
version	int		JDBC driver property: version.

dataSource > recoveryAuthData

Authentication data for transaction recovery.

false

Attribute name	Data type	Default value	Description
password	Reversably encoded password (string)		Password of the user to use when connecting to the EIS. The value can be stored in clear text or encoded form. It is recommended that you encode the password. To do so, use the securityUtility tool with the encode option.
user	string		Name of the user to use when connecting to the EIS.

Trust Association Interceptor (trustAssociation)

Controls the operation of the trust association interceptor (TAI).

- interceptors
 - library
 - file
 - fileset
 - folder
 - properties

Attribute name	Data type	Default value	Description
failOverToAppAuthType	boolean	false	Allow an interceptor to fall back to the application authentication mechanism.
id	string		A unique configuration ID.
invokeForUnprotectedURI	boolean	false	Controls whether the TAI is invoked for an unprotected URI.

interceptors

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
className	string		Fully-qualified package name of the interceptor class.
enabled	boolean	true	Enables or disables the interceptor.
id	string		A unique configuration ID.
invokeAfterSSO	boolean	true	Invoke an interceptor after single sign-on (SSO).
invokeBeforeSSO	boolean	false	Invoke an interceptor before single sign-on (SSO).
libraryRef	A reference to top level library element (string).		A reference to the ID of the shared library configuration.

interceptors > library

A reference to the ID of the shared library configuration.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.

Attribute name	Data type	Default value	Description
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
name	string		Name of shared library for administrators

interceptors > library > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

interceptors > library > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

interceptors > library > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

interceptors > properties

Collection of properties for the interceptor.

false

User Information (userInfo)

Specifies the user information that is included in the response of the openID provider.

Attribute name	Data type	Default value	Description
alias	string	email	Specifies an alias name.
count	int Minimum: 1	1	Specifies how much userInfo is included in the response of the openID provider.
id	string		A unique configuration ID.
required	boolean	true	Specifies whether user information is required or not.
uriType	string	http://axschema.org/contact/email	Specifies a URI type.

Variable Declaration (variable)

Declare a new variable by specifying the name and value for the variable.

Attribute name	Data type	Default value	Description
name	string		The name of the variable.
value	string		The value to be assigned to the variable.

Virtual Host (virtualHost)

A virtual host provides a logical grouping for configuring web applications to a particular host name. The default virtual host (default_host) is suitable for most simple configurations.

- allowFromEndpoint
 - accessLogging
 - httpOptions
 - sslOptions
 - tcpOptions
- hostAlias

Attribute name	Data type	Default value	Description
allowFromEndpointRef	List of references to top level httpEndpoint elements (comma-separated string).		Specify the identifier of one or more HTTP endpoints to restrict inbound traffic for this virtual host to the specified endpoints.
enabled	boolean	true	Enable this virtual host.
id	string		A unique configuration ID.

allowFromEndpoint

Specify the identifier of one or more HTTP endpoints to restrict inbound traffic for this virtual host to the specified endpoints.

false

Attribute name	Data type	Default value	Description
accessLoggingRef	A reference to top level httpAccessLogging element (string).		HTTP access logging configuration for the endpoint.
enabled	boolean	true	Toggle the availability of an endpoint. When true, this endpoint will be activated by the dispatcher to handle HTTP requests.
host	string	localhost	IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource. Use '*' for all available network interfaces.

Attribute name	Data type	Default value	Description
httpOptionsRef	A reference to top level httpOptions element (string).	defaultHttpOptions	HTTP protocol options for the endpoint.
httpPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests. Use -1 to disable this port.
httpsPort	int Minimum: -1 Maximum: 65535		The port used for client HTTP requests secured with SSL (https). Use -1 to disable this port.
id	string		A unique configuration ID.
onError	<ul style="list-style-type: none"> • IGNORE • FAIL • WARN 	WARN	<p>Action to take after a failure to start an endpoint.</p> <p>IGNORE Server will not issue any warning and error messages when it incurs a configuration error.</p> <p>FAIL Server will issue a warning or error message on the first error occurrence and then stop the server.</p> <p>WARN Server will issue warning and error messages when it incurs a configuration error.</p>
sslOptionsRef	A reference to top level sslOptions element (string).		SSL protocol options for the endpoint.
tcpOptionsRef	A reference to top level tcpOptions element (string).	defaultTCPOptions	TCP protocol options for the endpoint.

allowFromEndpoint > accessLogging

HTTP access logging configuration for the endpoint.

false

Attribute name	Data type	Default value	Description
enabled	boolean	true	Enable access logging.
filePath	Path to a file	\${server.output.dir}/logs/http_access.log	Directory path and name of the access log file. Standard variable substitutions, such as \${server.output.dir}, can be used when specifying the directory path.

Attribute name	Data type	Default value	Description
logFormat	string	%h %u %{t}W "%r" %s %b	Specifies the log format that is used when logging client access information.
maxFileSize	int Minimum: 0	20	Maximum size of a log file, in megabytes, before being rolled over; a value of 0 means no limit.
maxFiles	int Minimum: 0	2	Maximum number of log files that will be kept, before the oldest file is removed; a value of 0 means no limit.

allowFromEndpoint > httpOptions

HTTP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
keepAliveEnabled	boolean	true	Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.
maxKeepAliveRequests	int Minimum: -1	100	Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.
persistTimeout	A period of time with second precision	30s	Amount of time that a socket will be allowed to remain idle between requests. This setting only applies if persistent connections are enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
readTimeout	A period of time with second precision	60s	Amount of time to wait for a read request to complete on a socket after the first read occurs. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
removeServerHeader	boolean	false	Removes server implementation information from HTTP headers and also disables the default Liberty profile welcome page.
writeTimeout	A period of time with second precision	60s	Amount of time to wait on a socket for each portion of the response data to be transmitted. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

allowFromEndpoint > sslOptions

SSL protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
sessionTimeout	A period of time with second precision	1d	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
sslRef	A reference to top level ssl element (string).		The default SSL configuration repertoire. The default value is defaultSSLSettings.
suppressHandshakeErrors	boolean	false	Disable logging of SSL handshake errors. SSL handshake errors can occur during normal operation, however these messages can be useful when SSL is behaving unexpectedly.

allowFromEndpoint > tcpOptions

TCP protocol options for the endpoint.

false

Attribute name	Data type	Default value	Description
inactivityTimeout	A period of time with millisecond precision	60s	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
soReuseAddr	boolean	true	Enables immediate rebind to a port with no active listener.

hostAlias

Associate a host and port with this virtual host, using the host:port syntax. The specified host can be an IP address, domain name server (DNS) hostname with a domain name suffix, the DNS hostname, or * for a wildcard match on all hostnames. Note that IPv6 addresses must be enclosed in [].

false

string

Web Container Application Security (webAppSecurity)

Configures web container application security.

Attribute name	Data type	Default value	Description
allowAuthenticationFailOverToAuthMethod	<ul style="list-style-type: none"> • FORM • BASIC 		<p>Specifies the authentication fail over method that will be used when the certificate authentication fails. Valid values are BASIC and FORM.</p> <p>FORM %allowAuthenticationFailOverToAuthMethod</p> <p>BASIC %allowAuthenticationFailOverToAuthMethod</p>
allowFailOverToBasicAuth	boolean	false	<p>Specifies whether to fail over to basic authentication when certificate authentication fails. The equivalent custom property in the full application server profile is com.ibm.wsspi.security.web.failOverToBasicAuth</p>
allowLogoutPageRedirectToAnyHost	boolean	false	<p>Warning, security risk: Setting this property to true may open your systems to potential URL redirect attacks. If set to true, any host can be specified for the logout page redirect. If set to false, and the logout page points to a different host, or one not listed in the logout page redirect domain list, then a generic logout page is displayed. The equivalent custom property in the full application server profile is com.ibm.websphere.security.allowAnyLogoutE</p>

Attribute name	Data type	Default value	Description
displayAuthenticationRealm	boolean	false	Warning, security risk: if this property is set to true, and the user registry's realm name contains sensitive information, it is displayed to the user. For example, if an LDAP configuration is used, the LDAP server hostname and port are displayed. This configuration controls what the HTTP basic authentication login window displays when the realm name is not defined in the application web.xml. If the realm name is defined in the application web.xml file, this property is ignored. If set to true, the realm name displayed will be the user registry realm name for the LTPA authentication mechanism. If set to false, the realm name displayed will be "Default Realm". The equivalent custom property in the full application server profile is com.ibm.websphere.security.displayRealm.
httpOnlyCookies	boolean	true	Specifies whether the HTTP only (HttpOnly) cookies option is enabled.
includePathInWASReqURL	boolean	false	Setting the Path parameter can allow the client/browser to manage multiple WASReqURL cookies during multiple concurrent logins on the same user agent. The equivalent custom property in the full application server profile is com.ibm.websphere.security.setContextRoot.
loginFormURL	string		Specifies the global URL of a form login page including the root context. The form login page must be part of the WAR file. If a form login application does not specify the form login page in the web.xml file, it uses the global form login URL.

Attribute name	Data type	Default value	Description
logoutOnHttpSessionExpire	boolean	false	Specifies whether users will be logged out after the HTTP session timer expires. If set to false, the user credential will stay active until the Single Sign-On token timeout occurs. The equivalent custom property in the full application server profile is <code>com.ibm.ws.security.web.logoutOnHTTPSession</code>
logoutPageRedirectDomainNames	String		A pipe () separated list of domain names that are allowed for the logout page redirect (localhost is implied). The equivalent custom property in the full application server profile is <code>com.ibm.websphere.security.logoutExitPageDom</code>
postParamCookieSize	int	16384	Size of the POST parameter cookie. If the size of the cookie is larger than the browser limit, unexpected behavior may occur. The value of this property must be a positive integer and represents the maximum size of the cookie in bytes. The equivalent custom property in the full application server profile is <code>com.ibm.websphere.security.util.postParamMax</code>
postParamSaveMethod	<ul style="list-style-type: none"> • Cookie • Session • None 	Cookie	<p>Specifies where POST parameters are stored upon redirect. Valid values are cookie (POST parameters are stored in a cookie), session (POST parameters are stored in the HTTP Session) and none (POST parameters are not preserved). The equivalent custom property in the full application server profile is <code>com.ibm.websphere.security.util.postParamSave</code></p> <p>Cookie Cookie</p> <p>Session Session</p> <p>None None</p>

Attribute name	Data type	Default value	Description
preserveFullyQualifiedReferrer	boolean	false	Warning, security risk: Setting this to true may open your systems to potential URL redirect attacks. This property specifies whether the fully qualified referrer URL for form login redirects is preserved. If false, the host for the referrer URL is removed and the redirect is to localhost. The equivalent custom property in the full application server profile is <code>com.ibm.websphere.security.util.fullyQualifiedReferrer</code> .
singleSignonEnabled	boolean	true	Specifies whether single sign-on is enabled.
ssoCookieName	string	LtpaToken2	Customizes the SSO cookie name. A custom cookie name allows you to logically separate authentication between SSO domains and to enable customized authentication to a particular environment. Before setting this value, consider that setting a custom cookie name can cause an authentication failure. For example, a connection to a server that has a custom cookie property set sends this custom cookie to the browser. A subsequent connection to a server that uses either the default cookie name or a different cookie name, is not able to authenticate the request via a validation of the in-bound cookie. The equivalent custom property in the full application server profile is <code>com.ibm.websphere.security.customSSOCookieName</code> .
ssoDomainNames	string		A pipe () separated list of domain names that SSO Cookies should be presented. The equivalent custom property in the full application server profile is <code>com.ibm.ws.security.config.SingleSignonConfig</code> .

Attribute name	Data type	Default value	Description
ssoRequiresSSL	boolean	false	Specifies whether a SSO cookie is sent over SSL. The equivalent property in the full application server profile is requiresSSL.
ssoUseDomainFromURL	boolean	false	Specifies whether to use the domain name from the request URL for the cookie domain.
trackLoggedOutSSOCookies	boolean	false	Specifies whether to track LTPA single signon tokens that are logged out on a server so that it can not be reused on the same server.
useAuthenticationDataForUnprotectedResource	boolean	true	Specifies whether authentication data can be used when accessing an unprotected resource. The unprotected resource can access validated authenticated data that it previously could not access. This option enables the unprotected resource to call the getRemoteUser, isUserInRole, and getUserPrincipal methods to retrieve an authenticated identity. The equivalent custom property in the full application server profile is com.ibm.wsspi.security.webAuthReq=pers
useOnlyCustomCookieName	boolean	false	Specifies whether to use only the custom cookie name.
wasReqURLRedirectDomainNames	String		A pipe (!) separated list of domain names that are allowed for the WASReqURL page redirect. The hostname found on the form login request is implied.
webAlwaysLogin	boolean	false	Specifies whether the login() method will throw an exception when an identity has already been authenticated.

Web Application (webApplication)

Defines the properties of a web application.

- application-bnd
 - security-role
 - group

- run-as
- special-subject
- user
- classloader
 - commonLibrary
 - file
 - fileset
 - folder
 - privateLibrary
 - file
 - fileset
 - folder

Attribute name	Data type	Default value	Description
autoStart	boolean	true	Indicates whether or not the server automatically starts the application.
contextRoot	string		Context root of an application.
id	string		A unique configuration ID.
location	A file, directory or url.		Location of an application expressed as an absolute path or a path relative to the server-level apps directory.
name	string		Name of an application.
suppressUncoveredHttpMethodWarning	boolean	false	Option to suppress uncovered HTTP method warning message during application deployment.

application-bnd

Binds general deployment information included in the application to specific resources.

false

Attribute name	Data type	Default value	Description
version	string		Version of the application bindings specification.

application-bnd > security-role

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	string		Name of a security role.

application-bnd > security-role > group

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		Group access ID
id	string		A unique configuration ID.
name	string		Name of a group possessing a security role.

application-bnd > security-role > run-as

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
password	Reversably encoded password (string)		Password of a user required to access a bean from another bean. The value can be stored in clear text or encoded form. To encode the password, use the securityUtility tool with the encode option.
userid	string		ID of a user required to access a bean from another bean.

application-bnd > security-role > special-subject

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
type	<ul style="list-style-type: none">• EVERYONE• ALL_AUTHENTICATED_USERS		One of the following special subject types: ALL_AUTHENTICATED_USERS, EVERYONE. EVERYONE Everyone ALL_AUTHENTICATED_USERS All authenticated users

application-bnd > security-role > user

A unique configuration ID.

false

Attribute name	Data type	Default value	Description
access-id	string		A user access ID in the general form user:realmName/userUniqueId. A value will be generated if one is not specified.
id	string		A unique configuration ID.
name	string		Name of a user possessing a security role.

classloader

Defines the settings for an application classloader.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
classProviderRef	List of references to top level resourceAdapter elements (comma-separated string).		List of class provider references. When searching for classes or resources, this class loader will delegate to the specified class providers after searching its own class path.
commonLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are shared with other classloaders.
delegation	<ul style="list-style-type: none"> • parentFirst • parentLast 	parentFirst	<p>Controls whether parent classloader is used before or after this classloader. If parent first is selected then delegate to immediate parent before searching the classpath. If parent last is selected then search the classpath before delegating to the immediate parent.</p> <p>parentFirst Parent first</p> <p>parentLast Parent last</p>
privateLibraryRef	List of references to top level library elements (comma-separated string).		List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

classloader > commonLibrary

List of library references. Library class instances are shared with other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > commonLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > commonLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).

Attribute name	Data type	Default value	Description
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > commonLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

classloader > privateLibrary

List of library references. Library class instances are unique to this classloader, independent of class instances from other classloaders.

false

Attribute name	Data type	Default value	Description
apiTypeVisibility	string	spec,ibm-api,api	The types of API package this library's class loader will be able to see, as a comma-separated list of any combination of the following: spec, ibm-api, api, third-party.
description	string		Description of shared library for administrators
filesetRef	List of references to top level fileset elements (comma-separated string).		Id of referenced Fileset
id	string		A unique configuration ID.
name	string		Name of shared library for administrators

classloader > privateLibrary > file

Id of referenced File

false

Attribute name	Data type	Default value	Description
id	string		A unique configuration ID.
name	Path to a file		Fully qualified filename

classloader > privateLibrary > fileset

Id of referenced Fileset

false

Attribute name	Data type	Default value	Description
caseSensitive	boolean	true	Boolean to indicate whether or not the search should be case sensitive (default: true).
dir	Path to a directory	\${server.config.dir}	The base directory to search for files.
excludes	string		The comma or space separated list of file name patterns to exclude from the search results, by default no files are excluded.
id	string		A unique configuration ID.
includes	string	*	The comma or space separated list of file name patterns to include in the search results (default: *).
scanInterval	A period of time with millisecond precision	0	Scanning interval to check the fileset for changes as a long with a time unit suffix h-hour, m-minute, s-second, ms-millisecond (e.g. 2ms or 5s). Disabled (scanInterval=0) by default. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

classloader > privateLibrary > folder

Id of referenced folder

false

Attribute name	Data type	Default value	Description
dir	Path to a directory		Directory or folder to be included in the library classpath for locating resource files
id	string		A unique configuration ID.

Web Container (webContainer)

Configuration for the web container.

Attribute name	Data type	Default value	Description
allowExpressionFactoryPerApp	boolean	false	Toggle to load the ExpressionFactory that is set by the application. Enable this custom property if you are using a custom EL implementation (for example, JUEL) that needs to set its own ExpressionFactory.
allowIncludeSendError	boolean	false	Allow RequestDispatch to send errors on Include methods. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.allowincludesend
asyncMaxSizeTaskPool	int	5000	Maximum size of tasks in the Async task pool before automatically purging canceled tasks. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.asyncmaxsizetask
asyncPurgeInterval	int	30000	Time interval to wait between each required purge of the cancelled task pool. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.asyncpurgeinterval
asyncTimeoutDefault	int	30000	Async servlet timeout value used when a timeout value has not been explicitly specified. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.asynctimeoutdefa
asyncTimerThreads	int	2	Maximum number of threads to use for async servlet timeout processing. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.asynctimerthreads

Attribute name	Data type	Default value	Description
channelWriteType	string	async	When set to 'sync', responses will be written synchronously; otherwise, responses will be written asynchronously. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.channelwritetype.
copyAttributesKeySet	boolean	false	Web container will return an enumeration of a copy of the list of attributes to the servlet to avoid a concurrent access error by the servlet. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.copyattributeskeyset.
decodeUrlAsUtf8	boolean	true	Decode URLs using an encoding setting of UTF-8.
decodeUrlPlusSign	boolean	false	Decode the plus sign when it is part of the URL. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.decodeurlplussign.
defaultHeadRequestBehavior	boolean	false	Restore the behavior where the HEAD request is not subject to the security constraint defined for the GET method. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.DefaultHeadRequestBehavior.
defaultTraceRequestBehavior	boolean	false	Restore HTTP TRACE processing. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.DefaultTraceRequestBehavior.
deferServletLoad	boolean	true	Defer servlet loading and initialization until the first request.
deferServletRequestListenerDestroyOnError	boolean	false	Toggle if you want to defer ServletRequestListener destroy when there is an error serving the request. The default value is false. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.deferServletRequestListenerDestroyOnError.
directoryBrowsingEnabled	boolean	false	Enable directory browsing of an application.

Attribute name	Data type	Default value	Description
disableXPoweredBy	boolean	false	Disable setting the X-Powered-By header. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.disablexpoweredby
disallowAllFileServing	boolean	false	Disables all file serving by applications. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.disallowAllFileServing
disallowServeServletsByClassName	boolean	true	Disallows the use of serveServletsByClassNameEnabled on the application server level. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.disallowserveservletsbyclassname
dispatcherRethrowsError	boolean	true	Web container will re-throw errors allowing interested resources to process them. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.dispatcherRethrowsError
doNotServeByClassName	string		A semi-colon delimited list of classes to be completely disallowed from being served by classname. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.donotservebyclassname
emptyServletMappings	boolean	false	Toggle if you want to return an empty collection, instead of null, when no servlet mappings are added. The default value is false. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.emptyServletMappings
enableDefaultIsELIgnoredInTags	boolean	false	Always evaluate whether to ignore EL expressions in tag files. If parent JSP files have different isELIgnored settings, the setting will be re-evaluated in the tag file. The equivalent custom property in the full application server profile is com.ibm.ws.jsp.enabledefaultisignoredintags

Attribute name	Data type	Default value	Description
enableErrorExceptionTypeFirst	boolean	false	Web container is updated to search and use the exception-type before the error-code. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst
enableJspMappingOverride	boolean	false	Allow the JSP mapping to be overridden so that the application can serve the JSP contents itself. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.enablejspmappingoverride
enableMultiReadOfPostData	boolean	false	Retain post data for multiple read accesses. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.enablemultireadofpostdata
exposeWebInfOnDispatch	boolean	false	If true, a servlet can access files in the WEB-INF directory. If false (default), a servlet cannot access files the WEB-INF directory.
fileServingEnabled	boolean	true	Enable file serving if this setting was not explicitly specified for the application.
fileWrapperEvents	boolean	false	Web container will generate SMF and PMI data when serving the static files. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.fileWrapperEvents
httpsIndicatorHeader	string		For SSL offloading, set to the name of the HTTP header variable inserted by the SSL accelerator/proxy/load balancer.
ignoreSemiColonOnRedirectToWelcomePage	boolean	false	Toggle to ignore the trailing semi-colon when redirecting to the welcome page. The default value is false. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.ignoreSemiColonOnRedirectToWelcomePage

Attribute name	Data type	Default value	Description
ignoreSessiononStaticFileRequests	boolean	false	Improves performance by preventing the web container from accessing a session for static file requests involving filters. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequests
invokeFilterInitAtStartup	boolean	true	Web container will call the filter's init() method at application startup. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.invokeFilterInitAtStartup
listeners	string		A comma separated list of listener classes.
logServletContainerInitializationClassLoadingErrors	boolean	false	Log servlet container class loading errors as warnings rather than logging them only when debug is enabled. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.logServletContainerInitializationClassLoadingErrors
metaInfResourcesCacheSize	int	20	Initial size (number of entries) of the meta-inf resource cache. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.metaInfResourcesCacheSize
parseUtf8PostData	boolean	false	Web container will detect non URL encoded UTF-8 post data and include it in the parameter values. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.parseUtf8PostData
serveServletsByClassnameEnabled	boolean	false	Enable servlets to be accessed in a web application using a class name if not explicitly specified.
setContentLengthOnClose	boolean	true	Toggle to set content length when an application explicitly closes the response. The default value is true; however, set this value to false if an application response contains double-byte characters.

Attribute name	Data type	Default value	Description
skipMetaInfResourcesProcessing	boolean	false	Do not search the meta-inf directory for application resources. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.skipmetainfresources.
suppressHtmlRecursiveErrors	boolean	false	Suppresses the exception information from appearing in the HTML output when there is a recursive error that cannot be handled by an application's configured error page. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.suppressHtmlRecursiveErrors.
symbolicLinksCacheSize	int	1000	Initial size of the symbolic link cache. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.SymbolicLinksCacheSize.
tolerateSymbolicLinks	boolean	false	Enables the web container to support the use of symbolic links. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.TolerateSymbolicLinks.
useSemiColonAsDelimiterInURI	boolean	false	Toggle to use the semi-colon as a delimiter in the request URI. The default value is false. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.useSemiColonAsDelimiterInURI.
xPoweredBy	string		Alternative string for the X-Powered-By header setting. The equivalent custom property in the full application server profile is com.ibm.ws.webcontainer.xpoweredby. There is no default value for this property. If the property is not set, the value of the X-Powered-By header is set to Servlet/<servlet spec version>, as defined by the Servlet specification.

WAS WebSocket Outbound (wsocOutbound)

Configuration properties for WAS WebSocket outgoing connection requests.

- httpOptions

- sslOptions
- tcpOptions

Attribute name	Data type	Default value	Description
httpOptionsRef	A reference to top level httpOptions element (string).	defaultHttpOptions	HTTP protocol options for WAS WebSocket outbound
sslOptionsRef	A reference to top level sslOptions element (string).		SSL protocol options for WAS WebSocket outbound
tcpOptionsRef	A reference to top level tcpOptions element (string).	defaultTCPOptions	TCP protocol options for WAS WebSocket outbound

httpOptions

HTTP protocol options for WAS WebSocket outbound

false

Attribute name	Data type	Default value	Description
keepAliveEnabled	boolean	true	Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.
maxKeepAliveRequests	int Minimum: -1	100	Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.
persistTimeout	A period of time with second precision	30s	Amount of time that a socket will be allowed to remain idle between requests. This setting only applies if persistent connections are enabled. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
readTimeout	A period of time with second precision	60s	Amount of time to wait for a read request to complete on a socket after the first read occurs. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
removeServerHeader	boolean	false	Removes server implementation information from HTTP headers and also disables the default Liberty profile welcome page.
writeTimeout	A period of time with second precision	60s	Amount of time to wait on a socket for each portion of the response data to be transmitted. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

sslOptions

SSL protocol options for WAS WebSocket outbound

false

Attribute name	Data type	Default value	Description
sessionTimeout	A period of time with second precision	1d	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.

Attribute name	Data type	Default value	Description
sslRef	A reference to top level ssl element (string).		The default SSL configuration repertoire. The default value is defaultSSLSettings.
suppressHandshakeErrors	boolean	false	Disable logging of SSL handshake errors. SSL handshake errors can occur during normal operation, however these messages can be useful when SSL is behaving unexpectedly.

tcpOptions

TCP protocol options for WAS WebSocket outbound

false

Attribute name	Data type	Default value	Description
inactivityTimeout	A period of time with millisecond precision	60s	Amount of time to wait for a read or write request to complete on a socket. This value is overridden by protocol-specific timeouts. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.
soReuseAddr	boolean	true	Enables immediate rebind to a port with no active listener.

z/OS Logging (zosLogging)

Configuration properties for logging on z/OS.

Attribute name	Data type	Default value	Description
enableLogToMVS	boolean	false	Enable routing of USS-started server messages to the MVS console.
hardCopyMessage	string		A comma-separated list of message IDs to be written to the hardcopy log.
wtoMessage	string		A comma-separated list of message IDs to be written to operator.

Feature management

Features are the units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server.

Use the configuration file `server.xml` to declare which features you want to load. The set of features is enclosed within the `<featureManager>` element, and each feature within the `<feature>` subelement. For example:

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

You can specify any feature in the server configuration file. Some features include other features within them. The same feature can be included in one or more other features. At run time, the feature manager computes the combined list of content that is required to support the requested set of features.

For information about the main available features, see “Liberty features” on page 483. For information about the restrictions that apply to each feature, see “Runtime environment known issues and restrictions” on page 1480.

Dynamic changes to feature configuration

When you change the feature configuration, the feature manager recalculates the list of required bundles, stops and uninstalls those bundles that are no longer required, and installs and starts any additions. Therefore, all features are designed to cope with other features that are being dynamically added or removed.

8.5.5.4

Singleton Features

With the delivery of the first set of features for Java EE 7, there are now two versions of the same feature:

- `servlet-3.0`
- `servlet-3.1`

Unlike other application servers, you can choose which version of this feature to use in a server configuration. `servlet-3.0` preserves behavior for existing applications, while `servlet-3.1` provides new capabilities for new or modified applications. Although there is a choice of specification versions, no additional configuration properties are required, or provided, to control individual differences between the two versions.

The servlet feature is a singleton feature, which means that you can configure only one version for use in a server: either `servlet-3.0` or `servlet-3.1`. If you have applications that need different versions of the servlet feature, you must deploy them in different servers. Many other features include the servlet feature as a dependency. In the WebSphere Liberty product, these features have been updated to work with either version. This ensures that you can configure a complete “stack” of features when you use `servlet-3.1`, but features from other sources might not have been updated to “tolerate” `servlet-3.1`. To enable features to “tolerate” `servlet-3.1`, modify the feature manifest as follows:

```
Subsystem-Content: com.ibm.websphere.appserver.servlet-3.0; ibm.tolerates="3.1"; type="osgi.subsystem.feature"
```

If your server configuration includes multiple versions of a singleton feature, either through direct configuration in the `server.xml` file, or through feature dependencies, that configuration is in error and neither version of that feature is loaded. This error results in a message similar to the following:

```
[ERROR ] CWWKF0033E: The singleton features servlet-3.1 and servlet-3.0 cannot be loaded at the same time. The configured f
```

To resolve this problem, ensure that the configured features all specify, or tolerate, the same version of that singleton feature. If you have hard requirements on both feature versions, you must move some of your applications to a different server. For more information on tolerating singleton features, see *Tolerating singleton features*.

Superseded features

The **superseded** label on a feature indicates that a new feature or a combination of features might provide an advantage over using the superseded feature. For example, new, finer-grained features might be used in place of the superseded one in order to reduce the server footprint by excluding content that might not be necessary. The new feature or features might not completely replace the function of the superseded feature, so you must consider your scenario before deciding whether to change your configuration. Superseded features remain completely supported and valid for use in your configuration; the **superseded** label just provides an indication that you might be able to improve your configuration.

Very occasionally, a feature that includes other features is superseded by a new version of the feature that does not include all those features; the features that are not included in the new version are considered to be *separated*. If your application needs to use the functions of a separated feature, you must explicitly add the separated feature to your configuration.

For example, featureA and featureB have the following conditions:

- featureA-1.0 includes featureB-1.0
- featureA-2.0 does not include featureB-1.0 (or any later versions of featureB)

If your application uses featureB functions, either of these configurations will work:

- Include featureA-1.0 in your server.xml file
- Include featureA-2.0 and featureB-1.0 in your server.xml file

Liberty features

Features are the units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server.

The following table lists the Liberty features that are supported for each WebSphere Application Server Liberty edition.

Tip: To install all features that apply your Liberty edition, you can install a feature bundle add-on. WebSphere Application Server Network Deployment Liberty has two bundles: The `ndMemberBundle` add-on contains most features and is for servers that are clustered and auto-scaled in a Liberty collective, and the `ndControllerBundle` add-on contains a small set of features only for managing Liberty collectives.

Table 6. Liberty features supported for each WebSphere Application Server Liberty edition

Liberty feature	WebSphere Application Server Liberty Core	WebSphere Application Server and WebSphere Application Server - Express®	WebSphere Application Server Network Deployment (Distributed operating systems and IBM i)	WebSphere Application Server for z/OS®
Feature bundle add-on	libertyCoreBundle	baseBundle	ndMemberBundle: All except controller features ndControllerBundle: Only features marked with ¹	zosBundle
Java EE 7 Web Profile				
8.5.5.6 beanValidation-1.1	✓	✓	✓	✓
8.5.5.6 cdi-1.2	✓	✓	✓	✓
8.5.5.6 ejbLite-3.2	✓	✓	✓	✓
el-3.0	✓	✓	✓	✓
8.5.5.6 jaxrs-2.0	✓	✓	✓	✓
8.5.5.6 jaxrsClient-2.0	✓	✓	✓	✓
jdbc-4.1	✓	✓	✓	✓
jndi-1.0	✓	✓	✓	✓
8.5.5.6 jpa-2.1	✓	✓	✓	✓
8.5.5.6 jsf-2.2	✓	✓	✓	✓
jsonp-1.0	✓	✓	✓	✓
jsp-2.3	✓	✓	✓	✓
managedBeans-1.0	✓	✓	✓	✓
servlet-3.1	✓	✓	✓	✓
8.5.5.6 webProfile-7.0	✓	✓	✓	✓
websocket-1.0	✓	✓	✓	✓
websocket-1.1	✓	✓	✓	✓
Java EE 7 Full Platform				
8.5.5.6 appClientSupport-1.0		✓	✓	✓
8.5.5.6 appSecurityClient-1.0		✓	✓	✓
8.5.5.6 batch-1.0		✓	✓	✓
concurrent-1.0	✓	✓	✓	✓
8.5.5.6 ejb-3.2		✓	✓	✓
8.5.5.6 ejbHome-3.2		✓	✓	✓

Table 6. Liberty features supported for each WebSphere Application Server Liberty edition (continued)

Liberty feature	WebSphere Application Server Liberty Core	WebSphere Application Server and WebSphere Application Server - Express®	WebSphere Application Server Network Deployment (Distributed operating systems and IBM i)	WebSphere Application Server for z/OS®
8.5.5.6 ejbPersistentTimer-3.2		✓	✓	✓
8.5.5.6 ejbRemote-3.2		✓	✓	✓
8.5.5.6 j2eeManagement-1.1		✓	✓	✓
8.5.5.6 jacc-1.5	✓	✓	✓	✓
8.5.5.6 jaspic-1.1	✓	✓	✓	✓
8.5.5.6 javaee-7.0		✓	✓	✓
8.5.5.6 javaeeClient-7.0		✓	✓	✓
8.5.5.6 javaMail-1.5	✓	✓	✓	✓
jaxb-2.2		✓	✓	✓
jaxws-2.2		✓	✓	✓
8.5.5.6 jca-1.7		✓	✓	✓
jcaInboundSecurity-1.0		✓	✓	✓
8.5.5.6 jms-2.0		✓	✓	✓
8.5.5.6 jmsMdb-3.2		✓	✓	✓
8.5.5.6 mdb-3.2		✓	✓	✓
8.5.5.6 wasJmsClient-2.0		✓	✓	✓
wasJmsSecurity-1.0		✓	✓	✓
wasJmsServer-1.0		✓	✓	✓
Extended Programming Models				
cloudant-1.0		✓	✓	✓
couchdb-1.0		✓	✓	✓
distributedMap-1.0	✓	✓	✓	✓
json-1.0	✓	✓	✓	✓
mongodb-2.0		✓	✓	✓
8.5.5.7 rtcomm-1.0		✓	✓	✓
8.5.5.7 rtcommGateway-1.0		✓	✓	✓
8.5.5.7 sipServlet-1.1		✓	✓	✓
Enterprise OSGi				
blueprint-1.0	✓	✓	✓	✓

Table 6. Liberty features supported for each WebSphere Application Server Liberty edition (continued)

Liberty feature	WebSphere Application Server Liberty Core	WebSphere Application Server and WebSphere Application Server - Express®	WebSphere Application Server Network Deployment (Distributed operating systems and IBM i)	WebSphere Application Server for z/OS®
osgiAppIntegration-1.0	✓	✓	✓	✓
8.5.5.9 osgiBundle-1.0	✓	✓	✓	✓
osgi.jpa-1.0	✓	✓	✓	✓
wab-1.0	✓	✓	✓	✓
Operations				
8.5.5.8 apiDiscovery-1.0	✓	✓	✓	✓
appSecurity-1.0	✓	✓	✓	✓
appSecurity-2.0	✓	✓	✓	✓
8.5.5.6 batchManagement-1.0		✓	✓	✓
8.5.5.7 bells-1.0	✓	✓	✓	✓
8.5.5.9 bluemixUtility-1.0	✓	✓	✓	✓
8.5.5.6 eventLogging-1.0	✓	✓	✓	✓
ldapRegistry-3.0	✓	✓	✓	✓
localConnector-1.0	✓	✓	✓	✓
8.5.5.9 logstashCollector-1.0	✓	✓	✓	✓
monitor-1.0	✓	✓	✓	✓
oauth-2.0	✓	✓	✓	✓
openid-2.0	✓	✓	✓	✓
openidConnectClient-1.0	✓	✓	✓	✓
openidConnectServer-1.0	✓	✓	✓	✓
osgiConsole-1.0	✓	✓	✓	✓
8.5.5.9 passwordUtilities-1.0		✓	✓	✓
8.5.5.6 requestTiming-1.0	✓	✓	✓	✓
restConnector-1.0	✓	✓	✓	✓
8.5.5.7 samlWeb-2.0	✓	✓	✓	✓
serverStatus-1.0	✓	✓	✓	✓
sessionDatabase-1.0	✓	✓	✓	✓
spnego-1.0	✓	✓	✓	✓
ssl-1.0	✓	✓	✓	✓
timedOperations-1.0	✓	✓	✓	✓
webCache-1.0	✓	✓	✓	✓

Table 6. Liberty features supported for each WebSphere Application Server Liberty edition (continued)

Liberty feature	WebSphere Application Server Liberty Core	WebSphere Application Server and WebSphere Application Server - Express®	WebSphere Application Server Network Deployment (Distributed operating systems and IBM i)	WebSphere Application Server for z/OS®
8.5.5.6 wmqJmsClient-2.0		✓	✓	✓
wsSecurity-1.1		✓	✓	✓
8.5.5.9 wsAtomicTransaction-1.2		✓	✓	✓
Systems Management				
adminCenter-1.0	✓	✓	✓ ¹	✓
clusterMember-1.0			✓	✓
collectiveController-1.0			✓ ¹	✓
collectiveMember-1.0	✓	✓	✓	✓
dynamicRouting-1.0			✓ ¹	✓
8.5.5.7 healthAnalyzer-1.0			✓	✓
8.5.5.7 healthManager-1.0			✓ ¹	✓
scalingController-1.0			✓ ¹	✓
scalingMember-1.0			✓	✓
z/OS				
zosConnect-1.0				✓
8.5.5.7 zosConnect-1.2				✓
zosLocalAdapters-1.0				✓
zosSecurity-1.0				✓
zosTransaction-1.0				✓
zosWlm-1.0				✓
Java EE 6 Web Profile				
beanValidation-1.0	✓	✓	✓	✓
cdi-1.0	✓	✓	✓	✓
ejbLite-3.1	✓	✓	✓	✓
jdbc-4.0	✓	✓	✓	✓
jndi-1.0	✓	✓	✓	✓
jpa-2.0	✓	✓	✓	✓
jsf-2.0	✓	✓	✓	✓
jsp-2.2	✓	✓	✓	✓
servlet-3.0	✓	✓	✓	✓

Table 6. Liberty features supported for each WebSphere Application Server Liberty edition (continued)

Liberty feature	WebSphere Application Server Liberty Core	WebSphere Application Server and WebSphere Application Server - Express®	WebSphere Application Server Network Deployment (Distributed operating systems and IBM i)	WebSphere Application Server for z/OS®
webProfile-6.0	✓	✓	✓	✓
Java EE 6 Technologies				
jaxb-2.2		✓	✓	✓
jaxrs-1.1	✓	✓	✓	✓
jaxws-2.2		✓	✓	✓
jca-1.6		✓	✓	✓
jcaInboundSecurity-1.0		✓	✓	✓
jms-1.1		✓	✓	✓
jmsMdb-3.1		✓	✓	✓
mdb-3.1		✓	✓	✓
wasJmsClient-1.1		✓	✓	✓
wasJmsSecurity-1.0		✓	✓	✓
wasJmsServer-1.0		✓	✓	✓
wmqJmsClient-1.1		✓	✓	✓

Feature descriptions

The following list contains information about the features you can add to your server configuration. Including a feature in the configuration might cause one or more additional features to be loaded automatically. For example, if you include the `wab-1.0` feature, the `servlet-3.0` and `blueprint-1.0` features are loaded automatically. Each feature includes a brief description, and an example of how the feature is declared within the `<featureManager>` element inside the `server.xml` file. For example:

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

Java EE 7 Web Profile

8.5.5.6 Bean validation

```
<feature>beanValidation-1.1</feature>
```

The `beanValidation-1.1` feature provides validations for JavaBeans at each layer of an application. The validation can be applied to all layers of JavaBeans in an application by using annotations or a `validation.xml` deployment descriptor.

See “Bean validation feature restrictions” on page 1485.

For beanValidation-1.1 feature configuration information, see “Bean Validation 1.1” on page 513.

8.5.5.6 CDI

```
<feature>cdi-1.2</feature>
```

The cdi-1.2 feature enables support for the Contexts and Dependency Injection 1.2 specification on Liberty.

See “Administering Contexts and Dependency Injection applications on Liberty” on page 1071.

For cdi-1.2 feature configuration information, see “Contexts and Dependency Injection 1.2” on page 516.

Enterprise Java Beans (EJB) Lite

8.5.5.6 <feature>ejbLite-3.2</feature>

The ejbLite-3.2 feature provides support for EJB applications that are written to the EJB Lite subset of the EJB 3.2 specification.

Note that the EJB 3.2 Lite API Group does not include the embeddable EJB container, and the product does not provide an EJB 3.2 embeddable container.

Also, the following features are not compatible with the ejbLite-3.2 feature:

- cdi-1.0
- jmsMdb-3.1
- mdb-3.1

For ejbLite-3.2 feature configuration information, see “Enterprise JavaBeans Lite 3.2” on page 520.

Expression Language 3.0

```
<feature>el-3.0</feature>
```

This feature enables support for the Expression Language (EL) 3.0.

See “Configuring Liberty for Expression Language 3.0” on page 1067.

For el-3.0 feature configuration information, see “Expression Language 3.0” on page 521.

8.5.5.6 Java API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-2.0</feature>
```

The jaxrs-2.0 feature provides support for the Java API for RESTful Web Services on Liberty.

See “Configuring JAX-RS 2.0 client” on page 1351 and “JAX-RS 2.0 integration with EJB and CDI” on page 1360.

For jaxrs-2.0 feature configuration information, see “Java RESTful Services 2.0” on page 542.

8.5.5.6 Java EE Client API for JAX-RS 2.0

```
<feature>jaxrsClient-2.0</feature>
```

The jaxrsClient-2.0 feature provides support for Java Client API for JAX-RS 2.0

See “Configuring JAX-RS 2.0 client” on page 1351 and “JAX-RS 2.0 integration with EJB and CDI” on page 1360.

For jaxrsClient-2.0 feature configuration information, see “Java RESTful Services Client 2.0” on page 542.

Java Database Connectivity (JDBC)

```
<feature>jdbc-4.1</feature>
```

You can take an existing application that uses Java Database Connectivity (JDBC) and a data source, and deploy the application to a server. The `jdbc-4.1` feature provides support for applications that access a database.

See “Configuring relational database connectivity in Liberty” on page 1047.

For `jdbc-4.1` feature configuration information, see “Java Database Connectivity 4.1” on page 528.

Java Naming and Directory Interface (JNDI)

```
<feature>jndi-1.0</feature>
```

The `jndi-1.0` feature provides support for a single JNDI entry definition in the server configuration of Liberty.

See “Developing with the JNDI default namespace in a Liberty feature” on page 1139.

For `jndi-1.0` feature configuration information, see “Java Naming and Directory Interface” on page 531.

8.5.5.6 Java Persistence API 2.1

```
<feature>jpa-2.1</feature>
```

The `jpa-2.1` feature provides support for applications that use application-managed and container-managed JPA written to the JPA 2.1 specification and is backed by EclipseLink.

See “Java Persistence API (JPA) feature overview” on page 618.

For `jpa-2.1` feature configuration information, see “Java Persistence API 2.1” on page 534.

8.5.5.6 JavaServer Faces (JSF)

```
<feature>jsf-2.2</feature>
```

This feature enables support for web applications that use the Java Server Faces (JSF) 2.2 framework. This framework simplifies the construction of user interfaces.

See “Configuring Liberty for JavaServer Faces 2.2” on page 1068.

For `jsf-2.2` feature configuration information, see “JavaServer Faces 2.2” on page 552.

JavaScript Object Notation Processing

```
<feature>jsonp-1.0</feature>
```

The Java API for JSON Processing (JSON-P) feature provides a standardized method for constructing and manipulating data to be rendered in JavaScript Object Notation (JSON).

For `jsonp-1.0` feature configuration information, see “JavaScript Object Notation Processing” on page 549.

JavaServer Pages (JSP)

```
<feature>jsp-2.3</feature>
```

This feature enables support for Java Server Pages (JSPs) that are written to the JSP 2.3 specification. This framework simplifies the construction of user interfaces. Enabling this feature also enables the Expression Language (EL) version 3.0 feature. `el-3.0`.

See “Configuring Liberty for JavaServer Pages 2.3” on page 1070.

For `jsp-2.3` feature configuration information, see “JavaServer Pages 2.3” on page 554.

Managed Beans

```
<feature>managedBeans-1.0</feature>
```

The `managedBeans-1.0` feature provides support for the Managed Beans 1.0 specification (JSR-316). This feature enables use of the `javax.annotation.ManagedBean` annotation.

For `managedBeans-1.0` feature configuration information, see “Java EE Managed Bean 1.0” on page 528.

Servlet 3.1

```
<feature>servlet-3.1</feature>
```

The `servlet-3.1` feature enables support for HTTP Servlets that are written to the Java Servlet 3.1 specification.

See “Configuring Liberty for Servlet 3.1” on page 1059 and “Servlet 3.1 behavior changes” on page 1060.

For `servlet-3.1` feature configuration information, see “Java Servlets 3.1” on page 545.

8.5.5.6 Web Profile 7.0

```
<feature>webProfile-7.0</feature>
```

This feature provides a convenient combination of the Liberty features that are required to support the Java EE 7 Web Profile.

For `webProfile-7.0` feature configuration information, see “Java EE Web Profile 7.0” on page 530.

WebSocket

```
<feature>websocket-1.0</feature>
```

```
<feature>websocket-1.1</feature>
```

WebSocket is a standard protocol that enables a web browser or client application and a web server application to communicate by using one full duplex connection.

See Liberty: WebSocket and “Developing WebSocket applications in Liberty” on page 1326.

For `websocket-1.0` feature configuration information, see “Java WebSocket 1.0” on page 547.

For `websocket-1.1` feature configuration information, see “Java WebSocket 1.1” on page 548.

Java EE 7 Full Platform

8.5.5.6 Application Client Support

```
<feature>appClientSupport-1.0</feature>
```

The `appClientSupport-1.0` feature enables the server to process Java EE metadata inside the client module of an application, for example, read the deployment descriptor XML file and/or annotations and make them available to other modules in the application if necessary. It also enables the remote application client process to communicate with the server to do JNDI lookups.

The `appClientSupport-1.0` feature is enabled in the `server.xml` file only.

8.5.5.6 Application Client Container Security

```
<feature>appSecurityClient-1.0</feature>
```

To enable security on the client container, add the `appSecurityClient-1.0` feature to your `client.xml` file.

The `appSecurityClient-1.0` feature enables SSL, CSIV2, and JAAS on the client. You must configure SSL to ensure communications between the client and server are secure and encrypted.

8.5.5.6 Batch

```
<feature>batch-1.0</feature>
```

The `batch-1.0` feature enables the use of the JSR-352 programming model.

Managed Executors and Thread Factories

`<feature>concurrent-1.0</feature>`

The `concurrent-1.0` feature enables the creation of managed executor services that allow applications to submit tasks to run concurrently, with thread context that is managed by the application server. The feature also enables the creation of managed thread factories to create threads that run with the thread context of the component that looks up the managed thread factory.

See “Configuring managed scheduled executors” on page 1017.

For `concurrent-1.0` feature configuration information, see “Concurrency Utilities for Java EE 1.0” on page 514.

Enterprise Java Beans (EJB)

8.5.5.6 `<feature>ejb-3.2</feature>`

The `ejb-3.2` feature provides support for EJB applications that are written to the EJB 3.2 specification.

This feature includes the following features:

- **8.5.5.6** `<feature>ejbLite-3.2</feature>`

This feature gives support for EJB applications that are written to the EJB Lite subset of the EJB 3.2 specification. For `ejbLite-3.2` feature configuration information, see “Enterprise JavaBeans Lite 3.2” on page 520.

- **8.5.5.6** `<feature>ejbHome-3.2</feature>`

This feature gives support for the EJB 2.x APIs.

- **8.5.5.6** `<feature>ejbPersistentTimer-3.2</feature>`

This feature gives support for persistent EJB timers.

- **8.5.5.6** `<feature>ejbRemote-3.2</feature>`

This feature gives support for remote EJB interfaces.

- **8.5.5.6** `<feature>mdb-3.2</feature>`

This feature gives support for message-driven beans.

The `mdb-3.2` feature supersedes the `jmsMdb-3.2` feature.

If full EJB 3.2 support is not required, various combinations of these features can be used to provide the support that you need.

8.5.5.6 J2EE Management 1.1

`<feature>j2eeManagement-1.1</feature>`

The `j2eeManagement-1.1` feature provides standard interfaces to manageable aspects of Java EE 7 and enables applications to use the interfaces defined in the JSR 77 specification.

To invoke Management EJB APIs, the server configuration must have both the `j2eeManagement-1.1` and `ejbRemote-3.2` features in a feature manager. After both features are in the server configuration, you can invoke Management EJB API through JNDI name lookup. The Management EJB binding name (JNDI lookup name) is `ejb/mejb/MEJB`.

8.5.5.6 Java Authorization Contract for Containers 1.5

`<feature>jacc-1.5</feature>`

The `jacc-1.5` feature enables support for Java Authorization Contract for Containers (JACC) version 1.5. In order to add the `jacc-1.5` feature to your server, you need to add the third party JACC provider which is not a part of the WebSphere Application Server Liberty.

8.5.5.6 Java Authentication SPI for Containers 1.1

`<feature>jaspic-1.1</feature>`

The `jaspic-1.1` feature enables support for securing the server runtime environment and applications using Java Authentication SPI for Containers (JASPIC) providers as defined in JSR-196.

8.5.5.6 **Java EE**

```
<feature>javaee-7.0</feature>
```

This feature provides a convenient combination of the Liberty features that are required to support the Java EE 7.0 Full Platform.

8.5.5.6 **Java EE Application Client 7.0**

```
<feature>javaeeClient-7.0</feature>
```

This feature enables support for Java EE Application Client 7.0.

8.5.5.6 **JavaMail API**

```
<feature>javaMail-1.5</feature>
```

The JavaMail API supports communication between external mail servers and Liberty applications. See “Administering JavaMail on Liberty” on page 1074.

For `javaMail-1.5` feature configuration information, see “JavaMail 1.5” on page 548.

Java Architecture for XML Binding (JAXB)

```
<feature>jaxb-2.2</feature>
```

The `jaxb-2.2` feature provides support for the Java Architecture for XML Binding (JAXB) on Liberty.

See JAXB.

Java API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

The `jaxws-2.2` feature provides support for the Java API for XML-Based Web Services on Liberty.

- For web applications that support the JAX-WS programming model, you must enable the `servlet-3.0` and `jaxws-2.2` server features in the `server.xml` file.
- For EJB applications that support the JAX-WS programming model, you must enable the `ejbLite-3.1`, `servlet-3.0`, and `jaxws-2.2` server features in the `server.xml` file.
- For applications that use the global handler services, you must enable the `jaxrs-1.1` or the `jaxws-2.2` feature in the `server.xml` file.

8.5.5.6 **Java EE Connector Architecture 1.7**

```
<feature>jca-1.7</feature>
```

The `jca-1.7` feature provides configuration elements to define instances of connection factories, administered objects, and activation specifications, and to associate these instances with an installed resource adapter.

Java EE Connector Architecture Inbound Security

```
<feature>jcaInboundSecurity-1.0</feature>
```

The `jcaInboundSecurity-1.0` feature enables security inflow for resource adapters.

8.5.5.6 **Java Message Service 2.0**

```
<feature>jms-2.0</feature>
```

The `jms-2.0` feature enables the configuration of resource adapters to access messaging systems using the Java Message Service API. This also includes the configuration JMS connection factories, queues, topics and activation specifications. Any JMS resource adapter that complies with the JCA 1.6 specification can be used.

Embedded Liberty Messaging features

8.5.5.6 `<feature>wasJmsClient-2.0</feature>`

The `wasJmsClient-2.0` feature supersedes the `wasJmsClient-1.1` feature. The `wasJmsClient-2.0` feature is compliant with the JMS 2.0 specification and is supported only on IBM JDK 7 or later.

`<feature>wasJmsSecurity-1.0</feature>`

The `wasJmsSecurity-1.0` feature supports secure connections to the messaging engine. When the `wasJmsSecurity-1.0` feature is enabled, it starts authenticating and authorizing the users who are trying to connect to the messaging engine. The user is authenticated against the registry that is defined in the `server.xml` file. When the user wants to access a destination such as a topic or a queue, then the user must be granted the required permissions. The access to the destination is defined in the `<messagingSecurity>` element (the child element of the `messagingEngine` element) in the `server.xml` file. If the `wasJmsSecurity-1.0` feature is added and the `<messagingSecurity>` element is not defined in the `server.xml` file, then the users cannot connect to the messaging engine or perform any messaging action (for example, sending or receiving messages from the destinations).

Notes:

- Configuring the user registry is a prerequisite for the `wasJmsSecurity-1.0` feature. Ensure that a user registry is configured before the `wasJmsSecurity-1.0` feature is enabled.
- When you enable the `wasJmsSecurity-1.0` feature, you must also configure the `<messagingSecurity>` element, which is the child element of the `<messagingEngine>` element, in the `server.xml` file. This configuration enables authorized users to access messaging destinations.

`<feature>wasJmsServer-1.0</feature>`

The `wasJmsServer-1.0` feature enables the JMS messaging engine run time to be initialized. The messaging run time is responsible for providing the application connectivity, managing the state of destinations such as topics or queues, and handling quality of service, security, and transactions. This feature also provides support for the inbound connections from the remote messaging applications. The remote messaging applications can connect to the JMS messaging engine through TCP/IP over SSL or non-SSL.

To connect using SSL, you must enable the SSL feature.

Extended Programming Models

CouchDB

`<feature>couchdb-1.0</feature>`

The `couchdb-1.0` feature provides support for CouchDB instances and associated database connections. Access to CouchDB connections is available either by JNDI lookup or resource injection.

Cache Service

`<feature>distributedMap-1.0</feature>`

This feature provides a local cache service that can be accessed by using the DistributedMap API. A default cache is bound in JNDI at `services/cache/distributedmap`. You can distribute a cache by adding a network cache provider such as WebSphere eXtreme Scale.

For `distributedMap-1.0` feature configuration information, see “Distributed Map interface for Dynamic Caching” on page 518.

JavaScript Object Notation (JSON4J) Library

<feature>json-1.0</feature>

The json-1.0 feature provides access to the JSON4J library that provides a set of JSON handling classes for Java environments. The JSON4J library provides a simple Java model for constructing and manipulating data to be rendered as JSON data.

See Using JSON content in JAX-RS application requests and responses and JSON4J Libraries API.

For json-1.0 feature configuration information, see “JavaScript Object Notation for Java” on page 550.

MongoDB

<feature>mongodb-2.0</feature>

The mongodb-2.0 feature provides support for MongoDB instances and associated database connections. Access to MongoDB connections is available either by JNDI lookup or resource injection. The native com.mongodb API performs the database manipulation.

8.5.5.7 Real-Time Communications

<feature>rtcomm-1.0</feature>

The Liberty Real-Time Communications feature enables a highly-scalable call signalling engine that can be used to connect WebRTC clients into real-time audio/video/data calls. The feature supports both registration of clients as well as the exchange of signalling needed to create a WebRTC peer connection between two endpoints.

8.5.5.7 RTComm Gateway

<feature>rtcommGateway-1.0</feature>

The rtcommGateway-1.0 feature adds the capability for connecting Session Initiation Protocol (SIP) with RTComm WebRTC endpoints for the exchange of audio and video streams.

8.5.5.7 SIP Servlet

<feature>sipServlet-1.1</feature>

The sipServlet-1.1 feature provides support for SIP Servlet Specification 1.1, also known as JSR 289. Session Initiation Protocol (SIP) is a control protocol for many interactive services, including audio, video, and peer-to-peer communication.

Enterprise OSGi

Blueprint

<feature>blueprint-1.0</feature>

The blueprint-1.0 feature enables support for deploying OSGi applications that use the OSGi blueprint container specification. With the OSGi Applications support in WebSphere Application Server, you can develop and deploy modular applications that use Java EE and OSGi technologies.

See “Locating OSGi applications” on page 1138 and OSGi Applications.

For blueprint-1.0 feature configuration information, see “OSGi Blueprint” on page 560.

OSGi application integration

<feature>osgiAppIntegration-1.0</feature>

Use the osgiAppIntegration-1.0 feature to enable the OSGi applications that are available within the same Java virtual machine to share their services with each other.

For more information about Application-ImportService and Application-ExportService headers, see Application manifest files.

For `osgiAppIntegration-1.0` feature configuration information, see “OSGi Application Integration” on page 559.

8.5.5.9 OSGi bundle

```
<feature>osgiBundle-1.0</feature>
```

This feature enables support for deploying OSGi applications. With the OSGi Applications support, you can develop and deploy modular applications that use Java EE and OSGi technologies.

For `osgiBundle-1.0` feature configuration information, see `osgiBundle-1.0`.

OSGi JPA

```
<feature>osgi.jpa-1.0</feature>
```

The `osgi.jpa-1.0` feature provides JPA support for OSGi applications on Liberty.

See “Deploying OSGi applications to Liberty” on page 1338.

For `osgi.jpa-1.0` feature configuration information, see “OSGi Java Persistence API” on page 563.

Web application bundle (WAB)

```
<feature>wab-1.0</feature>
```

This feature enables support for warnings to be logged when certain operations in the application server are running more slowly than expected. The `wab-1.0` feature provides support for WABs that are inside enterprise bundles. The `wab-1.0` feature provides support for WABs that are inside enterprise bundles.

This feature supports the following resources packaged inside a WAB:

- Static web content and JSPs.
- HTTP servlets written to the Servlet 3.0 specification.
- Blueprint applications.

If you include the `wab-1.0` feature, you also include the `servlet-3.0` and `blueprint-1.0` features.

See “Deploying OSGi applications to Liberty” on page 1338.

For `wab-1.0` feature configuration information, see “OSGi Web Application Bundles” on page 564.

Operations

8.5.5.8

API Discovery

```
<feature>apiDiscovery-1.0</feature>
```

The `apiDiscovery-1.0` feature enables you to discover your REST API documentation. Use the feature to find REST APIs that are available on a Liberty server and then use the Swagger user interface to invoke the found REST endpoints. See “Discovering REST API documentation on a Liberty server” on page 1416.

Security

```
<feature>appSecurity-2.0</feature>
```

This version of the `appSecurity` feature provides only certain aspects of security, based explicitly on the presence of other features. Additionally, it does not automatically include the `servlet-3.0` or `ldapRegistry-3.0` features, thereby reducing the server footprint. To secure web applications, you must include the `servlet-3.0` feature. To enable EJB security, you must include the `ejbLite-3.1` feature. To support an LDAP user registry, you must include the `ldapRegistry-3.0` feature.

Note:

- The appSecurity-2.0 feature supersedes appSecurity-1.0. The features are the same except that appSecurity-2.0 does not automatically include servlet-3.0 or ldapRegistry-3.0. You can choose to use the appSecurity-2.0 version instead in your server configuration. See “Superseded features” on page 483.
 - To enable web security, you must specify the servlet-3.0 feature in the server.xml file.
 - To enable support for LDAP, you must specify the ldapRegistry-3.0 feature in the server.xml file.

The appSecurity-1.0 and appSecurity-2.0 features provide support for securing the server runtime environment and applications. The following aspects are supported:

- Basic user registry
- Lightweight Directory Access Protocol (LDAP) user registry
- Basic authorization
- Web application security
 - Basic authentication login
 - Form-login Form-logout
 - Programmatic APIs: getRemoteUser, getUserPrincipal, isUserInRole, authenticate, logout, and login.
- EJB application security
 - All security annotations and all security elements that can be specified in the ejb-jar.xml file.
 - Programmatic APIs: getCallerPrincipal, isCallerInRole, and getCallerIdentity. The getCallerIdentity API is not supported for Singleton session beans.
 - EJB extension settings in the ibm-ejb-jar-ext.xml file for **run-as-mode** of CALLER_IDENTITY and SPECIFIED_IDENTITY (SYSTEM_IDENTITY is not supported).

When you add the appSecurity-1.0 or appSecurity-2.0 feature to your server, you must also configure a user registry, such as the basic user registry or the LDAP user registry.

See Chapter 7, “Securing Liberty and its applications,” on page 1147 and “appSecurity-2.0 feature restrictions” on page 1483..

For appSecurity-1.0 feature configuration information, see “Application Security 1.0” on page 510.

For appSecurity-2.0 feature configuration information, see “Application Security 2.0” on page 510.

8.5.5.6 Managed Batch

```
<feature>batchManagement-1.0</feature>
```

The batchManagement-1.0 feature provides a REST interface for remote job submission and the batchManager command-line client utility.

8.5.5.7 Basic Extensions using Liberty Libraries (BELL)

```
<feature>bells-1.0</feature>
```

This feature enables the configuration of Basic Extensions using Liberty Libraries (BELL). Use this feature to extend some parts of the server runtime using libraries, rather than using Liberty features. BELL uses the Java Service Loader pattern to provide the implementation class name.

For bells-1.0 feature configuration information, see “Basic Extensions using Liberty Libraries” on page 511.

8.5.5.9 Bluemix Utility

```
<feature>bluemixUtility-1.0</feature>
```

This feature makes it easier to configure access to IBM Bluemix managed services. See “Setting up a Liberty server to use Bluemix services” on page 927.

For `bluemixUtility-1.0` feature configuration information, see `bluemixUtility-1.0`.

8.5.5.6 Event Logging

```
<feature>eventLogging-1.0</feature>
```

The `eventLogging-1.0` feature logs a record of events, such as the JDBC requests and servlet requests, and their durations.

See “Event Logging” on page 1467.

For `eventLogging-1.0` feature configuration information, see Event Logging.

ldapRegistry-3.0

```
<feature>ldapRegistry-3.0</feature>
```

The `ldapRegistry-3.0` feature provides support for LDAP user registry. The version 3.0 of the `ldapRegistry-3.0` feature is compliant with the LDAP Version 3 specifications. The `ldapRegistry-3.0` feature is not automatically enabled by the `appSecurity-2.0` feature. Using this feature, you can federate multiple LDAP repositories. Two or more LDAP repositories can be configured in the `server.xml` file, and you can get the consolidated results from multiple repositories for all LDAP operations.

For `ldapRegistry-3.0` feature configuration information, see “LDAP User Registry” on page 556.

Local JMX Connector

```
<feature>localConnector-1.0</feature>
```

The `localConnector-1.0` feature provides a local JMX connector that is built into the JVM. The JMX connector can be used only on the same host machine by someone running under the same user ID and the same JDK. It enables local access by JMX clients such as `jConsole`, or other JMX clients that use the Attach API.

See “Connecting to Liberty by using JMX” on page 1021.

For `localConnector-1.0` feature configuration information, see “JMX Local Connector” on page 522.

8.5.5.9 logstashCollector-1.0

```
<feature>logstashCollector-1.0</feature>
```

This feature gathers data from various sources and forwards the data to a Logstash server using Lumberjack protocol.

For `logstashCollector-1.0` feature configuration information, see `logstashCollector-1.0`.

Monitoring

```
<feature>monitor-1.0</feature>
```

The `monitor-1.0` feature provides Performance Monitoring Infrastructure (PMI) support on Liberty.

See Chapter 10, “Monitoring the Liberty server runtime environment,” on page 1423.

For `monitor-1.0` feature configuration information, see “Performance Monitoring” on page 566.

OAuth

```
<feature>oauth-2.0</feature>
```

The `oauth-2.0` feature provides support for securing access to resources using the OAuth 2.0 protocol.

For `oauth-2.0` feature configuration information, see “OAuth” on page 558.

OpenID

```
<feature>openid-2.0</feature>
```

This feature enables users to authenticate themselves to multiple entities without the need to manage multiple accounts or sets of credentials. Liberty supports OpenID 2.0 and plays a role as a Relying Party in web single-sign-on. Accessing various entities like websites often requires a unique account that is associated with each entity. OpenID enables a single set of credentials that are handled by an OpenID Provider to grant access to any number of entities that support OpenID. See “OpenID” on page 1189.

For `openid-2.0` feature configuration information, see “OpenID” on page 564.

OpenID Connect Client

```
<feature>openidConnectClient-1.0</feature>
```

This feature enables web applications to integrate OpenID Connect Client 1.0 for authenticating users instead of, or in addition to, the configured user registry. See “OpenID Connect” on page 1190.

For `openidConnectClient-1.0` feature configuration information, see “OpenID Connect Client” on page 565.

OpenID Connect Provider

```
<feature>openidConnectServer-1.0</feature>
```

This feature enables web applications to integrate OpenID Connect Server 1.0 for authenticating users instead of, or in addition to, the configured user registry. See “OpenID Connect” on page 1190.

For `openidConnectServer-1.0` feature configuration information, see “OpenID Connect Provider” on page 566.

OSGi Console

```
<feature>osgiConsole-1.0</feature>
```

This feature enables an OSGi console to aid the debugging of the runtime environment. It can be used to display information about bundles, packages, and services. This information can be useful when developing your own features for product extensions.

See “Using an OSGi console” on page 967.

For `osgiConsole-1.0` feature configuration information, see “OSGi Debug Console” on page 562.

8.5.5.9 Password Utilities

```
<feature>passwordUtilities-1.0</feature>
```

This feature enables support for obtaining AuthData from an application using security plug-points.

8.5.5.6 Request Timing

```
<feature>requestTiming-1.0</feature>
```

The `requestTiming-1.0` provides warnings and diagnostic information for the slow or hung requests.

See “Slow and hung request detection” on page 1469.

For `requestTiming-1.0` feature configuration information, see Request Timing.

REST connector 1.0

```
<feature>restConnector-1.0</feature>
```

The `restConnector-1.0` feature provides a secure JMX connector that can be used locally or remotely using any JDK. It enables remote access by JMX clients through a REST-based connector and requires SSL and basic user security configuration.

See “Connecting to Liberty by using JMX” on page 1021 and, for details on REST connectors, see “Configuring secure JMX connection to Liberty” on page 1022.

8.5.5.6 For information about using REST APIs to transfer files, see “File transfer” on page 1034. For information about using REST APIs to transfer files to and from collective controllers, collective members, and registered hosts of a collective, see “Transferring files in a Liberty collective” on page 1035. To use the REST APIs, you add the `restConnector-1.0` feature to the server configuration.

For `restConnector-1.0` feature configuration information, see “JMX REST Connector 1.0” on page 523.

8.5.5.7 SAML Web Browser SSO

```
<feature>samlWeb-2.0</feature>
```

The `samlWeb-2.0` feature enables web applications to delegate user authentication to a SAML identity provider instead of a configured user registry.

For `samlWeb-2.0` feature configuration information, see “SAML web single sign-on version 2.0” on page 568.

Server status

```
<feature>serverStatus-1.0</feature>
```

The `serverStatus-1.0` feature enables Liberty servers to automatically publish their status to WebSphere Application Server deployment managers and job managers that are aware of the server as a resource in their Job configuration. The known states are Started and Stopped.

See Submitting jobs to manage Liberty servers and Installing Liberty server resources using the job manager.

For `serverStatus-1.0` feature configuration information, see “Job Manager Integration” on page 555.

Session Persistence

```
<feature>sessionDatabase-1.0</feature>
```

The `sessionDatabase-1.0` feature provides session affinity and failover support on Liberty.

See “Configuring session persistence for Liberty” on page 990.

For `sessionDatabase-1.0` feature configuration information, see “Database Session Persistence” on page 517.

SPNEGO

```
<feature>spnego-1.0</feature>
```

This feature enables users to log in to the Microsoft Domain controller once and access protected applications on Liberty servers without getting prompted again.

For more information on configuring SPNEGO on the Liberty server, see “Configuring SPNEGO authentication in Liberty” on page 1195.

For `spnego-1.0` feature configuration information, see “Simple and Protected GSSAPI Negotiation Mechanism” on page 569.

Secure Sockets Layer (SSL)

```
<feature>ssl-1.0</feature>
```

The `ssl-1.0` feature provides support for Secure Sockets Layer (SSL) connections. To use the secure HTTPS listener, you must enable this feature. Liberty provides a dummy keystore and a dummy truststore, which are the same as those provided by previous versions of WebSphere Application Server. The secure HTTPS listener is not started unless the `ssl-1.0` feature is enabled. If the feature is unavailable, the HTTPS listener is stopped.

To specify the SSL certificates, add a pointer in the server.xml file; see “Securing communications in Liberty” on page 1151. To change the HTTPS port, set the <httpsPort> attribute of the <httpEndpoint> element in the server.xml file; see “Specifying Liberty bootstrap properties” on page 897.

For ssl-1.0 feature configuration information, see “Secure Socket Layer” on page 568.

Timed Operations

```
<feature>timedOperations-1.0</feature>
```

This feature enables support for warnings to be logged when certain operations in the application server are running more slowly than expected.

See “Timed operations and JDBC calls” on page 1466.

For timedOperations-1.0 feature configuration information, see “Timed Operations” on page 571.

Dynamic caching service

```
<feature>webCache-1.0</feature>
```

This feature enables local caching for web responses. It includes the Cache Service (distributedMap) feature and performs automatic caching of web application responses to improve response times and throughput. To customize the response caching, you can include a cache-spec.xml file in your applications. You can distribute the cache by adding a network cache provider such as WebSphere eXtreme Scale.

For webCache-1.0 feature configuration information, see “Web Response Cache” on page 571.

8.5.5.6 WebSphere MQ Messaging feature

```
<feature>wmqJmsClient-2.0</feature>
```

The wmqJmsClient-2.0 feature provides applications with access to message queues hosted on IBM MQ through the JMS 2.0 API.

Web services security

```
<feature>wsSecurity-1.1</feature>
```

The wsSecurity-1.1 feature provides support for securing web services at the message level. To secure web services messages, you must enable this feature and the appSecurity-2.0 and jaxws-2.2 features. Web services security policies defined in a WSDL file are ignored and are not enforced unless the wsSecurity-1.1 feature is enabled.

8.5.5.9 Web Services Atomic Transaction

```
<feature>wsAtomicTransaction-1.2</feature>
```

The wsAtomicTransaction is an interoperable transaction protocol. It enables you to flow distributed transactions by using Web service messages, and coordinate in an interoperable manner between heterogeneous transaction infrastructures.

For wsAtomicTransaction-1.2 configuration information in Liberty, see Web Services Atomic Transaction in Liberty.

Systems Management

Administrative Center

```
<feature>adminCenter-1.0</feature>
```

The adminCenter-1.0 feature is a web-based graphical interface for managing Liberty servers and applications and other resources from a web browser on a cell phone, tablet, or computer.

See “Administering Liberty using Admin Center” on page 1074 and “Admin Center feature restrictions” on page 1484.

For adminCenter-1.0 feature configuration information, see “Admin Center” on page 508.

Cluster member

```
<feature>clusterMember-1.0</feature>
```

To add a member to a collective, add the clusterMember-1.0 feature and optionally `<clusterMember name="cluster_name"/>` to its server configuration. The cluster name is published to the controller, and this server becomes part of the specified cluster.

Collective controller

```
<feature>collectiveController-1.0</feature>
```

The collectiveController-1.0 feature enables controller functionality for a management collective and includes a management repository MBean that is accessible using the JMX/REST connector that is provided by the restConnector-1.0 feature. The collective controller acts as a storage and collaboration mechanism to which collective members can connect. The collectiveController-1.0 feature includes a ServerCommandMbean that can be used to remote start or stop servers that are managed by the collective controller. The collectiveController-1.0 feature and its capabilities are available only in IBMWebSphere Application Server Liberty Network Deployment. The feature is not available in IBM WebSphere Application Server Liberty, IBM WebSphere Application Server Liberty - Express, or IBM WebSphere Application Server Liberty Core.

Collective member

```
<feature>collectiveMember-1.0</feature>
```

The collectiveMember-1.0 feature enables a server to be a member of a management collective, allowing it to be managed by the collective controller.

See “Setting up the server-management environment for Liberty by using collectives” on page 907.

For collectiveMember-1.0 feature configuration information, see “Collective Member” on page 514.

Dynamic Routing

```
<feature>dynamicRouting-1.0</feature>
```

The Intelligent Management feature of the WebSphere plugin for Apache and IHS provides On Demand Router (ODR) capabilities for the plugin. This feature enables a server to run the dynamic routing service. The plugin can then connect to the ODR in order to dynamically route to all servers in the liberty collective.

8.5.5.7

Health Analyzer

```
<feature>healthAnalyzer-1.0</feature>
```

The Health Analyzer feature provides health data collection for the health manager for the Intelligent Management feature Health Management. The health analyzer feature provides monitoring services to a member server. It registers as an analytics handler, collects the necessary stats (PMI, HealthCenter) via the analytics collector and analyses the conditions.

8.5.5.7

Health Manager

```
<feature>healthManager-1.0</feature>
```

The Health Manager feature provides health monitoring and automatic actions based on health policies for the Intelligent Management feature Health Management. The health

manager feature embodies the core functions of health management. Selecting this feature will also enable the default condition plugins. This feature requires the presence of the `collectiveController` feature.

Scaling Controller

```
<feature>scalingController-1.0</feature>
```

The `scalingController-1.0` feature makes scaling decisions for Liberty. Multiple servers can run the Scaling Controller feature for high availability purposes. Only one server is actively making scaling decisions at any time. If that server is stopped, another server that is running the Scaling Controller feature can take over making scaling decisions.

Scaling Member

```
<feature>scalingMember-1.0</feature>
```

The `scalingMember-1.0` feature can be added to the **featureManagement** element of the `server.xml` of servers that are collective members. This will enable auto clustering of the collective members and will allow the servers to dynamically start/stop based on criteria specified by the scaling policy. This feature works in conjunction with the scaling controller feature. The scaling controller feature should be enabled in the collective controllers that are part of the collective.

z/OS

z/OS Connect

```
<feature>zosConnect-1.0</feature>
```

The `zosConnect-1.0` feature is service that encapsulates calling z/OS target applications using REST calls.

8.5.5.7 z/OS Connect 1.2

```
<feature>zosConnect-1.2</feature>
```

This feature provides a gateway between mobile, cloud, and web clients and z/OS back-end systems, such as CICS Transaction Server, IMS, and batch applications. It provides RESTful APIs and enables you to route HTTP requests to remote REST endpoints. It also accepts and returns JSON payloads and communicates with back-end systems by providing a data transformation service that converts JSON payloads to/from byte arrays consumable by z/OS native-language applications written in Cobol, PL/I, and C.

z/OS optimized local adapters

```
<feature>zosLocalAdapters-1.0</feature>
```

The `zosLocalAdapters-1.0` feature enables high-performance calling between native-language applications on z/OS and business logic in a Liberty server environment.

z/OS security

```
<feature>zosSecurity-1.0</feature>
```

The `zosSecurity-1.0` feature provides support on the z/OS platform for basic interactions with the SAF Registry, including authenticating users, and retrieving users, groups, or groups associated with users, from the SAF Registry.

z/OS transaction management

```
<feature>zosTransaction-1.0</feature>
```

Specifying this feature enables the application server to synchronize and appropriately manage transactional activity between the Resource Recovery Services (RRS), the transaction manager of the application server, and the resource manager.

z/OS workload management

```
<feature>zosWlm-1.0</feature>
```

The `zosWlm-1.0` feature provides access to z/OS native workload management (WLM) services.

Java EE 6 Web Profile

Bean validation

`<feature>beanValidation-1.0</feature>`

The `beanValidation-1.0` feature provides validations for JavaBeans at each layer of an application. The validation can be applied to all layers of JavaBeans in an application by using annotations or a `validation.xml` deployment descriptor.

See “Bean validation feature restrictions” on page 1485.

For `beanValidation-1.0` feature configuration information, see “Bean Validation 1.0” on page 512.

CDI

`<feature>cdi-1.0</feature>`

The `cdi-1.0` feature enables support for the Contexts and Dependency Injection 1.0 specification on Liberty.

See “Administering Contexts and Dependency Injection applications on Liberty” on page 1071.

For `cdi-1.0` feature configuration information, see “Contexts and Dependency Injection 1.0” on page 515.

Enterprise JavaBeans (EJB) Lite subset

`<feature>ejbLite-3.1</feature>`

The `ejbLite-3.1` feature provides support for EJB applications written to the EJB Lite subset of the EJB 3.1 specification.

The following functions are supported:

- An EJB module packaged in an EAR file.
- EJBs packaged in a WAR file.
- The `@Stateful`, `@Stateless`, `@Singleton`, and `@EJB` annotations.
- The `javax.annotation.security` annotations.
- Injection of JPA `EntityManager`, `EntityManagerFactory`, and JDBC `DataSource` into all types of session bean types.
- `ejb-jar.xml`.
- EJB interceptors.
- No-Interface View.
- Bean managed transactions (`UserTransaction`).

See “ejbLite-3.1 feature restrictions” on page 1485.

For `ejbLite-3.1` feature configuration information, see “Enterprise JavaBeans Lite 3.1” on page 519.

Java Database Connectivity (JDBC)

`<feature>jdbc-4.0</feature>`

You can take an existing application that uses Java Database Connectivity (JDBC) and a data source, and deploy the application to a server. The `jdbc-4.0` feature provides support for applications that access a database.

See “Deploying an existing JDBC application to Liberty” on page 1339.

For `jdbc-4.0` feature configuration information, see “Java Database Connectivity 4.0” on page 527.

Java Naming and Directory Interface (JNDI)

`<feature>jndi-1.0</feature>`

The `jndi-1.0` feature provides support for a single JNDI entry definition in the server configuration of Liberty.

For `jndi-1.0` feature configuration information, see “Java Naming and Directory Interface” on page 531.

Java Persistence API (JPA)

`<feature>jpa-2.0</feature>`

The `jpa-2.0` feature provides support for applications that use application-managed and container-managed JPA written to the JPA 2.0 specification. The support is built on Apache OpenJPA with extensions to support the container-managed programming model.

Extended Persistence Context is now available for use with Stateful Session beans.

See “Deploying a JPA application to Liberty” on page 1347.

For `jpa-2.0` feature configuration information, see “Java Persistence API 2.0” on page 532.

JavaServer Faces (JSF)

`<feature>jsf-2.0</feature>`

The `jsf-2.0` feature provides support for web applications that use the JSF framework. This framework simplifies the construction of user interfaces. If you include the `jsf-2.0` feature, you also include the `jsp-2.2` feature, because the JSF framework is an extension of the JSP framework.

For `jsf-2.0` feature configuration information, see “JavaServer Faces 2.0” on page 551.

JavaServer Pages (JSP)

`<feature>jsp-2.2</feature>`

If you include the `jsf-2.0` feature, you also include the `jsp-2.2` feature, because the JSF framework is an extension of the JSP framework. If you include the `jsp-2.2` feature, you also include the `servlet-3.0` feature.

See “jsp-2.2 feature restrictions” on page 1486.

For `jsp-2.2` feature configuration information, see “JavaServer Pages 2.2” on page 553.

Servlet 3.0

`<feature>servlet-3.0</feature>`

The `servlet-3.0` feature provides support for HTTP Servlets written to the Java Servlet 3.0 specification.

See Chapter 7, “Securing Liberty and its applications,” on page 1147.

For `servlet-3.0` feature configuration information, see “Java Servlets 3.0” on page 543.

Web Profile

`<feature>webProfile-6.0</feature>`

This feature provides a convenient combination of the Liberty features that are required to support the Java EE 6.0 Web Profile.

For `webProfile-6.0` feature configuration information, see “Java EE Web Profile 6.0” on page 529.

Java EE 6 Technologies

Java Architecture for XML Binding (JAXB)

`<feature>jaxb-2.2</feature>`

The jaxb-2.2 feature provides support for the Java Architecture for XML Binding (JAXB) on Liberty.

See JAXB.

Java API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-1.1</feature>
```

The jaxrs-1.1 feature provides support for the Java API for RESTful Web Services on Liberty.

- For EJB applications that use the jaxrs-1.1 server feature, you must enable the ejbLite-3.1 feature in the server.xml file.
- For JAX-RS applications that use CDI, you must enable the cdi-1.0 feature in the server.xml file.
- For applications that use the global handler services, you must enable the jaxrs-1.1 or the jaxws-2.2 feature in the server.xml file.

For jaxrs-1.1 feature configuration information, see “Java RESTful Services 1.1” on page 539.

Java API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

The jaxws-2.2 feature provides support for the Java API for XML-Based Web Services on Liberty.

- For web applications that support the JAX-WS programming model, you must enable the servlet-3.0 and jaxws-2.2 server features in the server.xml file.
- For EJB applications that support the JAX-WS programming model, you must enable the ejbLite-3.1, servlet-3.0, and jaxws-2.2 server features in the server.xml file.
- For applications that use the global handler services, you must enable the jaxrs-1.1 or the jaxws-2.2 feature in the server.xml file.

Java EE Connector Architecture

```
<feature>jca-1.6</feature>
```

The jca-1.6 feature provides configuration elements to define instances of connection factories, administered objects, and activation specifications, and to associate these instances with an installed resource adapter.

Java EE Connector Architecture Inbound Security

```
<feature>jcaInboundSecurity-1.0</feature>
```

The jcaInboundSecurity-1.0 feature enables security inflow for resource adapters.

Java Message Service 1.1

```
<feature>jms-1.1</feature>
```

The jms-1.1 feature enables the configuration of resource adapters to access messaging systems using the Java Message Service API. This also includes the configuration JMS connection factories, queues, topics, and activation specifications. Any JMS resource adapter that complies with the JCA 1.6 specification can be used.

Message-Driven beans

```
<feature>jmsMdb-3.1</feature>
```

The jmsMdb-3.1 feature provides support for deploying and configuring the JMS resources that are required for the message-driven beans (MDB) to run within Liberty. This feature enables MDB to interact with either the embedded Liberty messaging or WebSphere MQ.

Message-Driven Beans 3.1

```
<feature>mdb-3.1</feature>
```

The mdb-3.1 feature enables the use of Message-Driven Enterprise JavaBeans. MDBs allow asynchronous processing of messages within a Java EE component.

Embedded Liberty Messaging features

```
<feature>wasJmsClient-1.1</feature>
```

The wasJmsClient-1.1 feature enables support for JMS resource configurations (such as the connection factories, activation specifications, and queue and topic resources) and also provides the client libraries that are required by the messaging applications to connect to the JMS server on Liberty.

```
<feature>wasJmsSecurity-1.0</feature>
```

The wasJmsSecurity-1.0 feature supports secure connections to the messaging engine. When the wasJmsSecurity-1.0 feature is enabled, it starts authenticating and authorizing the users who are trying to connect to the messaging engine. The user is authenticated against the registry that is defined in the server.xml file. When the user wants to access a destination such as a topic or a queue, then the user must be granted the required permissions. The access to the destination is defined in the <messagingSecurity> element (the child element of the messagingEngine element) in the server.xml file. If the wasJmsSecurity-1.0 feature is added and the <messagingSecurity> element is not defined in the server.xml file, then the users cannot connect to the messaging engine or perform any messaging action (for example, sending or receiving messages from the destinations).

Notes:

- Configuring the user registry is a prerequisite for the wasJmsSecurity-1.0 feature. Ensure that a user registry is configured before the wasJmsSecurity-1.0 feature is enabled.
- When you enable the wasJmsSecurity-1.0 feature, you must also configure the <messagingSecurity> element, which is the child element of the <messagingEngine> element, in the server.xml file. This configuration enables authorized users to access messaging destinations.

```
<feature>wasJmsServer-1.0</feature>
```

The wasJmsServer-1.0 feature enables the JMS messaging engine run time to be initialized. The messaging run time is responsible for providing the application connectivity, managing the state of destinations such as topics or queues, and handling quality of service, security, and transactions. This feature also provides support for the inbound connections from the remote messaging applications. The remote messaging applications can connect to the JMS messaging engine through TCP/IP over SSL or non-SSL.

To connect using SSL, you must enable the SSL feature.

WebSphere MQ Messaging feature

```
<feature>wmqJmsClient-1.1</feature>
```

The wmqJmsClient-1.1 feature enables applications to use JMS messaging that connects to a IBM MQ server.

API Discovery 1.0

Enables discovery and exposure of REST APIs within Liberty.

Enabling this feature

To enable the API Discovery 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>apiDiscovery-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the API Discovery 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.apiDiscovery-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- json-1.0 - JavaScript Object Notation for Java
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

Third party API packages provided by this feature

- io.swagger.annotations

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the API Discovery 1.0 feature:

- administrator-role
- apiDiscovery
- authCache
- authentication
- authorization-roles
- basicRegistry
- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- mimeTypees
- quickStartSecurity
- tcpOptions
- trustAssociation
- virtualHost

Admin Center

The `adminCenter-1.0` feature enables the Liberty Admin Center, a web-based graphical interface for deploying, monitoring and managing Liberty servers in standalone and collective environments.

Enabling this feature

To enable the Admin Center feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>adminCenter-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Admin Center feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.adminCenter-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `distributedMap-1.0` - Distributed Map interface for Dynamic Caching
- `json-1.0` - JavaScript Object Notation for Java
- `jsp-2.2` - JavaServer Pages 2.2
- `jsp-2.3` - JavaServer Pages 2.3
- `restConnector-1.0` - JMX REST Connector 1.0
- `restConnector-2.0` - JMX REST Connector 2.0
- `servlet-3.0` - Java Servlets 3.0
- `servlet-3.1` - Java Servlets 3.1
- `ssl-1.0` - Secure Socket Layer

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Admin Center feature:

- `administrator-role`
- `authCache`
- `authentication`
- `authorization-roles`
- `basicRegistry`
- `channelfw`
- `classloading`
- `httpAccessLogging`
- `httpDispatcher`
- `httpEncoding`
- `httpEndpoint`
- `httpOptions`
- `httpProxyRedirect`
- `jaasLoginContextEntry`
- `jaasLoginModule`
- `library`
- `ltpa`
- `mimeTypes`
- `quickStartSecurity`
- `tcpOptions`
- `trustAssociation`
- `virtualHost`

Application Security 1.0

This feature is superseded by appSecurity-2.0. Support for securing the server runtime environment and applications. This feature enables servlet-3.0 and web application security, support for LDAP and basic user registries, and SSL. To support secure EJB applications, you must add the ejbLite-3.1 feature. When you add this feature to your server, you need to configure a user registry, such as the basic user registry or the LDAP user registry.

Enabling this feature

To enable the Application Security 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>appSecurity-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Application Security 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.appSecurity-1.0; type="osgi.subsystem.feature"
```

Features that this feature is superseded by

- ldapRegistry-3.0 - LDAP User Registry
- servlet-3.0 - Java Servlets 3.0
- appSecurity-2.0 - Application Security 2.0

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- ldapRegistry-3.0 - LDAP User Registry
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Application Security 2.0

This feature enables support for securing the server runtime environment and applications; it includes a basic user registry. This feature supersedes appSecurity-1.0 and does not include servlet-3.0 or support for the LDAP user registry. To secure web applications, add the servlet-3.0 feature. To secure EJB applications, add the ejbLite-3.1 feature. To use LDAP, add the ldapRegistry-3.0 feature. When you add the appSecurity-2.0 feature to your server, you need to configure a user registry, such as the basic user registry or the LDAP user registry.

Enabling this feature

To enable the Application Security 2.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>appSecurity-2.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Application Security 2.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.appSecurity-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- ssl-1.0 - Secure Socket Layer

Features that enable this feature

- appSecurity-1.0 - Application Security 1.0
- constrainedDelegation-1.0 - Kerberos Constrained Delegation for SPNEGO
- jacc-1.5 - Java Authorization Contract for Containers 1.5
- jaspic-1.1 - Java Authentication SPI for Containers 1.1
- oauth-2.0 - OAuth
- openid-2.0 - OpenID
- samlWeb-2.0 - SAML web single sign-on version 2.0
- spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism
- webProfile-6.0 - Java EE Web Profile 6.0
- webProfile-7.0 - Java EE Web Profile 7.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Application Security 2.0 feature:

- administrator-role
- authCache
- authentication
- basicRegistry
- classloading
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- quickStartSecurity

Basic Extensions using Liberty Libraries

This feature enables the configuration of Basic Extensions using Liberty Libraries (BELL).

Enabling this feature

To enable the Basic Extensions using Liberty Libraries feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>bells-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Basic Extensions using Liberty Libraries feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.bells-1.0; type="osgi.subsystem.feature"
```

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Basic Extensions using Liberty Libraries feature:

- bell
- classloading
- library

Bean Validation 1.0

The Bean Validation 1.0 specification provides an annotation based model for validating JavaBeans. It can be used to assert and maintain the integrity of data as it travels through an application.

Enabling this feature

To enable the Bean Validation 1.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>beanValidation-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Bean Validation 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.beanValidation-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- jpa-2.0 - Java Persistence API 2.0
- jsf-2.0 - JavaServer Faces 2.0
- osgi.jpa-1.0 - OSGi Java Persistence API
- webProfile-6.0 - Java EE Web Profile 6.0

Standard API packages provided by this feature

- javax.validation
- javax.validation.bootstrap
- javax.validation.constraints
- javax.validation.groups
- javax.validation.metadata
- javax.validation.spi

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Bean Validation 1.0 feature:

- classloading
- library

Bean Validation 1.1

The Bean Validation 1.1 specification provides an annotation based model for validating JavaBeans. It can be used to assert and maintain the integrity of data as it travels through an application.

Enabling this feature

To enable the Bean Validation 1.1 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>beanValidation-1.1</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Bean Validation 1.1 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.beanValidation-1.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- el-3.0 - Expression Language 3.0

Features that enable this feature

- jpa-2.0 - Java Persistence API 2.0
- jsf-2.0 - JavaServer Faces 2.0
- osgi.jpa-1.0 - OSGi Java Persistence API
- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.validation
- javax.validation.bootstrap
- javax.validation.constraints
- javax.validation.constraintvalidation
- javax.validation.executable
- javax.validation.groups
- javax.validation.metadata
- javax.validation.spi

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Bean Validation 1.1 feature:

- classloading
- library
- transaction

Bluemix Utilities 1.0

The Bluemix Utility feature can be used to quickly and easily configure access to IBM Bluemix managed services.

Enabling this feature

To enable the Bluemix Utilities 1.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>bluemixUtility-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Bluemix Utilities 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.bluemixUtility-1.0; type="osgi.subsystem.feature"
```

Collective Member

This feature enables a server to be a member of a management collective.

Enabling this feature

To enable the Collective Member feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>collectiveMember-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Collective Member feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.collectiveMember-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

SPI packages provided by this feature

- com.ibm.wsspi.collective.repository
- com.ibm.wsspi.collective.repository.publisher

Feature configuration elements

You can use the following elements in your server.xml file to configure the Collective Member feature:

- collectiveCertificate
- collectiveMember
- hostAuthInfo
- hostSingleton

Concurrency Utilities for Java EE 1.0

This feature enables the creation of managed executors that allow applications to submit tasks to run concurrently, with thread context that is managed by the application server. It also enables the creation of managed thread factories to create threads that run with the thread context of the component that looks up the managed thread factory.

Enabling this feature

To enable the Concurrency Utilities for Java EE 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>concurrent-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Concurrency Utilities for Java EE 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.concurrent-1.0; type="osgi.subsystem.feature"
```

Standard API packages provided by this feature

- javax.enterprise.concurrent

Feature configuration elements

You can use the following elements in your server.xml file to configure the Concurrency Utilities for Java EE 1.0 feature:

- classloading
- contextService
- managedExecutorService
- managedScheduledExecutorService
- managedThreadFactory

Contexts and Dependency Injection 1.0

The Contexts and Dependency Injection specification makes it easier to integrate Java EE components of different types. It provides a common mechanism to inject component such as EJBs or Managed Beans into other components such as JSPs or other EJBs.

Enabling this feature

To enable the Contexts and Dependency Injection 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>cdi-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Contexts and Dependency Injection 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.cdi-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- webProfile-6.0 - Java EE Web Profile 6.0

Standard API packages provided by this feature

- javax.decorator
- javax.enterprise.context
- javax.enterprise.context.spi
- javax.enterprise.event
- javax.enterprise.inject
- javax.enterprise.inject.spi
- javax.enterprise.util
- javax.inject
- javax.interceptor

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Contexts and Dependency Injection 1.0 feature:

- `cdiContainer`
- `classloading`
- `transaction`

Contexts and Dependency Injection 1.2

The Contexts and Dependency Injection specification makes it easier to integrate Java EE components of different types. It provides a common mechanism to inject component such as EJBs or Managed Beans into other components such as JSPs or other EJBs.

Enabling this feature

To enable the Contexts and Dependency Injection 1.2 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>cdi-1.2</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Contexts and Dependency Injection 1.2 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.cdi-1.2; type="osgi.subsystem.feature"
```

Features that enable this feature

- microProfile-1.0 - Micro Profile 1.0
- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.decorator
- javax.enterprise.context

- javax.enterprise.context.spi
- javax.enterprise.event
- javax.enterprise.inject
- javax.enterprise.inject.spi
- javax.enterprise.util
- javax.inject
- javax.interceptor

Third party API packages provided by this feature

- org.jboss.weld.context
- org.jboss.weld.context.api
- org.jboss.weld.context.beanstore
- org.jboss.weld.context.bound
- org.jboss.weld.context.conversation

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Contexts and Dependency Injection 1.2 feature:

- application
- applicationManager
- applicationMonitor
- cdi12
- classloading
- javaPermission
- library
- transaction

Database Session Persistence

This feature enables persistence of HTTP sessions to a datasource using JDBC. Persisting HTTP session data to a database allows recovery of the data after a server restart or unexpected server failure. Failover of HTTP sessions can be achieved by configuring multiple servers to persist data to the same location

Enabling this feature

To enable the Database Session Persistence feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>sessionDatabase-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Database Session Persistence feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.sessionDatabase-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- jdbc-4.0 - Java Database Connectivity 4.0
- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Database Session Persistence feature:

- classloading
- httpSession
- httpSessionDatabase
- transaction

Distributed Map interface for Dynamic Caching

This feature provides a local cache service which can be accessed through the DistributedMap API. A default cache is bound in JNDI at "services/cache/distributedmap". Caches can be distributed through addition of a network cache provider such as WebSphere eXtreme Scale.

Enabling this feature

To enable the Distributed Map interface for Dynamic Caching feature, add the following element declaration inside the featureManager element in your `server.xml` file:

```
<feature>distributedMap-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Distributed Map interface for Dynamic Caching feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.distributedMap-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- jndi-1.0 - Java Naming and Directory Interface

Features that enable this feature

- adminCenter-1.0 - Admin Center
- apiDiscovery-1.0 - API Discovery 1.0
- oauth-2.0 - OAuth
- openidConnectClient-1.0 - OpenID Connect Client
- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- samlWeb-2.0 - SAML web single sign-on version 2.0
- scim-1.0 - System for Cross-domain Identity Management
- webCache-1.0 - Web Response Cache

IBM API packages provided by this feature

- com.ibm.websphere.cache
- com.ibm.websphere.cache.exception
- com.ibm.websphere.exception
- com.ibm.ws.cache.spi
- com.ibm.wsspi.cache

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Distributed Map interface for Dynamic Caching feature:

- classloading
- distributedMap
- library

Enterprise JavaBeans Lite 3.1

This feature enables support for Enterprise JavaBeans written to the EJB Lite subset of the EJB 3.1 specification.

Enabling this feature

To enable the Enterprise JavaBeans Lite 3.1 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>ejbLite-3.1</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Enterprise JavaBeans Lite 3.1 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.ejbLite-3.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- jndi-1.0 - Java Naming and Directory Interface

Features that enable this feature

- webProfile-6.0 - Java EE Web Profile 6.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Enterprise JavaBeans Lite 3.1 feature:

- application
- applicationManager
- applicationMonitor
- classloading
- ejbApplication

- `ejbContainer`
- `enterpriseApplication`
- `javaPermission`
- `library`
- `transaction`
- `webApplication`

Enterprise JavaBeans Lite 3.2

This feature enables support for Enterprise JavaBeans written to the EJB Lite subset of the EJB 3.2 specification.

Enabling this feature

To enable the Enterprise JavaBeans Lite 3.2 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>ejbLite-3.2</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Enterprise JavaBeans Lite 3.2 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.ejbLite-3.2; type="osgi.subsystem.feature"
```

Features that this feature enables

- `jndi-1.0` - Java Naming and Directory Interface

Features that enable this feature

- `webProfile-7.0` - Java EE Web Profile 7.0

IBM API packages provided by this feature

- `com.ibm.websphere.ejbcontainer.mbean`

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Enterprise JavaBeans Lite 3.2 feature:

- `application`
- `applicationManager`
- `applicationMonitor`
- `classloading`
- `ejbApplication`
- `ejbContainer`
- `enterpriseApplication`
- `javaPermission`
- `library`
- `transaction`

- webApplication

Event Logging

Logs a record of events, such as JDBC requests and servlet requests, and their durations.

Enabling this feature

To enable the Event Logging feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>eventLogging-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Event Logging feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.eventLogging-1.0; type="osgi.subsystem.feature"
```

SPI packages provided by this feature

- com.ibm.wsspi.event.logging

Feature configuration elements

You can use the following elements in your server.xml file to configure the Event Logging feature:

- eventLogging

Expression Language 3.0

This feature enables support for the Expression Language (EL) 3.0.

Enabling this feature

To enable the Expression Language 3.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>el-3.0</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Expression Language 3.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.el-3.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- beanValidation-1.1 - Bean Validation 1.1
- jsp-2.3 - JavaServer Pages 2.3
- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.el

Federated User Registry

This feature enables support for federation of multiple user registries.

Enabling this feature

To enable the Federated User Registry feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>federatedRegistry-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Federated User Registry feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.federatedRegistry-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- ssl-1.0 - Secure Socket Layer

Features that enable this feature

- ldapRegistry-3.0 - LDAP User Registry
- scim-1.0 - System for Cross-domain Identity Management

SPI packages provided by this feature

- com.ibm.wsspi.security.wim
- com.ibm.wsspi.security.wim.exception
- com.ibm.wsspi.security.wim.model

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Federated User Registry feature:

- federatedRepository

JMX Local Connector

This feature allows the use of a local JMX connector that is built into the JVM to access JMX resources in the server. The JMX connector can only be used on the same host machine by a client that has the same user ID and the same JDK as the server process. It enables local access by JMX clients such as jConsole, or other JMX clients that use the Attach API.

Enabling this feature

To enable the JMX Local Connector feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>localConnector-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the JMX Local Connector feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.localConnector-1.0; type="osgi.subsystem.feature"
```

JMX REST Connector 1.0

A secure JMX connector that can be used locally or remotely using any JDK. It enables remote access by JMX clients via a REST-based connector and requires SSL and basic user security configuration. This feature is superseded by the restConnector-2.0 feature. This feature enables the jaxrs-1.1 feature.

Enabling this feature

To enable the JMX REST Connector 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>restConnector-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JMX REST Connector 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.restConnector-1.0; type="osgi.subsystem.feature"
```

Features that this feature is superseded by

- jaxrs-1.1 - Java RESTful Services 1.1
- json-1.0 - JavaScript Object Notation for Java
- restConnector-2.0 - JMX REST Connector 2.0

Features that this feature enables

- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- json-1.0 - JavaScript Object Notation for Java
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

Features that enable this feature

- adminCenter-1.0 - Admin Center
- collectiveMember-1.0 - Collective Member

IBM API packages provided by this feature

- com.ibm.websphere.filetransfer
- com.ibm.websphere.jmx.connector.rest
- com.ibm.ws.jmx.connector.client.rest

SPI packages provided by this feature

- com.ibm.wsspi.collective.plugins
- com.ibm.wsspi.collective.plugins.helpers

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the JMX REST Connector 1.0 feature:

- administrator-role
- authCache
- authentication
- authorization-roles
- basicRegistry
- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- mimeTypees
- quickStartSecurity
- remoteFileAccess
- tcpOptions
- trustAssociation
- virtualHost

JMX REST Connector 2.0

A secure JMX connector that can be used locally or remotely using any JDK. It enables remote access by JMX clients via a REST-based connector and requires SSL and basic user security configuration. This feature supersedes the `restConnector-1.0` feature and does not include the `jaxrs-1.1` feature.

Enabling this feature

To enable the JMX REST Connector 2.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>restConnector-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JMX REST Connector 2.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.restConnector-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- json-1.0 - JavaScript Object Notation for Java
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

Features that enable this feature

- adminCenter-1.0 - Admin Center
- collectiveMember-1.0 - Collective Member

IBM API packages provided by this feature

- com.ibm.websphere.filetransfer
- com.ibm.websphere.jmx.connector.rest
- com.ibm.ws.jmx.connector.client.rest

SPI packages provided by this feature

- com.ibm.wsspi.collective.plugins
- com.ibm.wsspi.collective.plugins.helpers

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the JMX REST Connector 2.0 feature:

- administrator-role
- authCache
- authentication
- authorization-roles
- basicRegistry
- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- mimeTypeypes
- quickStartSecurity
- remoteFileAccess
- tcpOptions
- trustAssociation
- virtualHost

Java Authentication SPI for Containers 1.1

This feature enables support for securing the server runtime environment and applications using Java Authentication SPI for Containers (JASPIC) providers as defined in JSR-196

Enabling this feature

To enable the Java Authentication SPI for Containers 1.1 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jaspic-1.1</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Authentication SPI for Containers 1.1 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jaspic-1.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Standard API packages provided by this feature

- javax.security.auth.message
- javax.security.auth.message.callback
- javax.security.auth.message.config
- javax.security.auth.message.module

SPI packages provided by this feature

- com.ibm.wsspi.security.jaspi

Java Authorization Contract for Containers 1.5

This feature enables support for Java Authorization Contract for Containers (JACC) version 1.5 In order to add the jacc-1.5 feature to your server, you need to add the third party JACC provider which is not a part of the WebSphere Application Server Liberty profile.

Enabling this feature

To enable the Java Authorization Contract for Containers 1.5 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jacc-1.5</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Authorization Contract for Containers 1.5 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jacc-1.5; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0

Standard API packages provided by this feature

- javax.security.jacc

IBM API packages provided by this feature

- com.ibm.wsspi.security.authorization.jacc

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Authorization Contract for Containers 1.5 feature:

- classloading

Java Database Connectivity 4.0

This feature enables the configuration of `DataSources` to access Databases from applications. Any database that complies with the JDBC 4.0 specification can be used; customized configuration of many specific providers is included. High performance connection pooling is also provided.

Enabling this feature

To enable the Java Database Connectivity 4.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jdbc-4.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Database Connectivity 4.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jdbc-4.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- jndi-1.0 - Java Naming and Directory Interface

Features that enable this feature

- jpa-2.0 - Java Persistence API 2.0
- osgi.jpa-1.0 - OSGi Java Persistence API
- sessionDatabase-1.0 - Database Session Persistence
- webProfile-6.0 - Java EE Web Profile 6.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Database Connectivity 4.0 feature:

- authData
- classloading

- connectionManager
- dataSource
- jdbcDriver
- library
- transaction

Java Database Connectivity 4.1

This feature enables the configuration of DataSources to access Databases from applications. Any JDBC driver that complies with the JDBC 4.1, 4.0, 3.0, or 2.x specification can be used; customized configuration of many specific providers is included. High performance connection pooling is also provided.

Enabling this feature

To enable the Java Database Connectivity 4.1 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jdbc-4.1</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Database Connectivity 4.1 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jdbc-4.1; type="osgi.subsystem.feature"
```

Features that enable this feature

- jpa-2.0 - Java Persistence API 2.0
- jpa-2.1 - Java Persistence API 2.1
- osgi.jpa-1.0 - OSGi Java Persistence API
- sessionDatabase-1.0 - Database Session Persistence
- webProfile-6.0 - Java EE Web Profile 6.0
- webProfile-7.0 - Java EE Web Profile 7.0

Feature configuration elements

You can use the following elements in your server.xml file to configure the Java Database Connectivity 4.1 feature:

- authData
- classloading
- connectionManager
- dataSource
- jdbcDriver
- library
- transaction

Java EE Managed Bean 1.0

This feature enables support for the Managed Beans 1.0 specification. Managed Beans provide a common foundation for different Java EE components types that are managed by a container. Common services provided to Managed Beans include resource injection, lifecycle management and the use of interceptors.

Enabling this feature

To enable the Java EE Managed Bean 1.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>managedBeans-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java EE Managed Bean 1.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.managedBeans-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `jndi-1.0` - Java Naming and Directory Interface

Features that enable this feature

- `webProfile-6.0` - Java EE Web Profile 6.0
- `webProfile-7.0` - Java EE Web Profile 7.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java EE Managed Bean 1.0 feature:

- `classloading`
- `ejbContainer`
- `library`
- `transaction`

Java EE Web Profile 6.0

This feature provides a convenient combination of the Liberty features that are required to support the Java EE Web Profile.

Enabling this feature

To enable the Java EE Web Profile 6.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>webProfile-6.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java EE Web Profile 6.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.webProfile-6.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `appSecurity-2.0` - Application Security 2.0
- `beanValidation-1.0` - Bean Validation 1.0

- cdi-1.0 - Contexts and Dependency Injection 1.0
- ejbLite-3.1 - Enterprise JavaBeans Lite 3.1
- jdbc-4.0 - Java Database Connectivity 4.0
- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface
- jpa-2.0 - Java Persistence API 2.0
- jsf-2.0 - JavaServer Faces 2.0
- jsp-2.2 - JavaServer Pages 2.2
- managedBeans-1.0 - Java EE Managed Bean 1.0
- servlet-3.0 - Java Servlets 3.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java EE Web Profile 6.0 feature:

- classloading
- transaction

Java EE Web Profile 7.0

This feature combines the Liberty features that support the Java EE 7.0 Web Profile.

Enabling this feature

To enable the Java EE Web Profile 7.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>webProfile-7.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java EE Web Profile 7.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.webProfile-7.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- beanValidation-1.1 - Bean Validation 1.1
- cdi-1.2 - Contexts and Dependency Injection 1.2
- ejbLite-3.2 - Enterprise JavaBeans Lite 3.2
- el-3.0 - Expression Language 3.0
- jaxrs-2.0 - Java RESTful Services 2.0
- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface
- jpa-2.1 - Java Persistence API 2.1
- jsf-2.2 - JavaServer Faces 2.2
- jsonp-1.0 - JavaScript Object Notation Processing
- jsp-2.3 - JavaServer Pages 2.3
- managedBeans-1.0 - Java EE Managed Bean 1.0
- servlet-3.1 - Java Servlets 3.1
- websocket-1.1 - Java WebSocket 1.1

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java EE Web Profile 7.0 feature:

- `classloading`
- `transaction`

Java Naming and Directory Interface

This feature enables the use of Java Naming and Directory Interface (JNDI) to access server configured resources such as DataSources or JMS Connection Factories. It also allows access to Java primitives configured in the server as a `jndiEntry`.

Enabling this feature

To enable the Java Naming and Directory Interface feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jndi-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Naming and Directory Interface feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jndi-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- `distributedMap-1.0` - Distributed Map interface for Dynamic Caching
- `ejbLite-3.1` - Enterprise JavaBeans Lite 3.1
- `ejbLite-3.2` - Enterprise JavaBeans Lite 3.2
- `jdbc-4.0` - Java Database Connectivity 4.0
- `jpa-2.0` - Java Persistence API 2.0
- `jpa-2.1` - Java Persistence API 2.1
- `managedBeans-1.0` - Java EE Managed Bean 1.0
- `osgi.jpa-1.0` - OSGi Java Persistence API
- `sessionDatabase-1.0` - Database Session Persistence
- `webProfile-6.0` - Java EE Web Profile 6.0
- `webProfile-7.0` - Java EE Web Profile 7.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Naming and Directory Interface feature:

- `classloading`
- `jndiEntry`
- `jndiObjectFactory`
- `jndiReferenceEntry`

- jndiURLEntry
- library

Java Persistence API 2.0

This feature enables support for applications that use application-managed and container-managed JPA written to the Java Persistence API 2.0 specification. The support is built on top of Apache OpenJPA with extensions to support the container-managed programming model.

Enabling this feature

To enable the Java Persistence API 2.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jpa-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Persistence API 2.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jpa-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- beanValidation-1.0 - Bean Validation 1.0
- beanValidation-1.1 - Bean Validation 1.1
- jdbc-4.0 - Java Database Connectivity 4.0
- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Features that enable this feature

- osgi.jpa-1.0 - OSGi Java Persistence API
- webProfile-6.0 - Java EE Web Profile 6.0

Standard API packages provided by this feature

- javax.persistence
- javax.persistence.criteria
- javax.persistence.metamodel
- javax.persistence.spi

Third party API packages provided by this feature

- org.apache.openjpa.abstractstore
- org.apache.openjpa.ant
- org.apache.openjpa.audit
- org.apache.openjpa.conf
- org.apache.openjpa.datacache
- org.apache.openjpa.ee
- org.apache.openjpa.enhance
- org.apache.openjpa.event
- org.apache.openjpa.instrumentation
- org.apache.openjpa.instrumentation.jmx

- org.apache.openjpa.jdbc.ant
- org.apache.openjpa.jdbc.conf
- org.apache.openjpa.jdbc.identifier
- org.apache.openjpa.jdbc.kernel
- org.apache.openjpa.jdbc.kernel.exps
- org.apache.openjpa.jdbc.meta
- org.apache.openjpa.jdbc.meta.strats
- org.apache.openjpa.jdbc.schema
- org.apache.openjpa.jdbc.sql
- org.apache.openjpa.kernel
- org.apache.openjpa.kernel.exps
- org.apache.openjpa.kernel.jpql
- org.apache.openjpa.lib.ant
- org.apache.openjpa.lib.conf
- org.apache.openjpa.lib.encryption
- org.apache.openjpa.lib.graph
- org.apache.openjpa.lib.identifier
- org.apache.openjpa.lib.instrumentation
- org.apache.openjpa.lib.jdbc
- org.apache.openjpa.lib.log
- org.apache.openjpa.lib.meta
- org.apache.openjpa.lib.rop
- org.apache.openjpa.lib.util
- org.apache.openjpa.lib.util.concurrent
- org.apache.openjpa.lib.util.svn
- org.apache.openjpa.lib.xml
- org.apache.openjpa.meta
- org.apache.openjpa.persistence
- org.apache.openjpa.persistence.criteria
- org.apache.openjpa.persistence.jdbc
- org.apache.openjpa.persistence.meta
- org.apache.openjpa.persistence.osgi
- org.apache.openjpa.persistence.query
- org.apache.openjpa.persistence.util
- org.apache.openjpa.persistence.validation
- org.apache.openjpa.slice
- org.apache.openjpa.slice.jdbc
- org.apache.openjpa.util
- org.apache.openjpa.validation
- org.apache.openjpa.xmlstore

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Persistence API 2.0 feature:

- `classloading`

- jpa
- library
- transaction

Java Persistence API 2.1

This feature enables support for applications that use application-managed and container-managed JPA written to the Java Persistence API 2.1 specification. The support is built on top of EclipseLink to support the container-managed programming model.

Enabling this feature

To enable the Java Persistence API 2.1 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jpa-2.1</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Persistence API 2.1 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jpa-2.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface

Features that enable this feature

- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.persistence
- javax.persistence.criteria
- javax.persistence.metamodel
- javax.persistence.spi

Third party API packages provided by this feature

- org.eclipse.persistence
- org.eclipse.persistence.annotations
- org.eclipse.persistence.config
- org.eclipse.persistence.core.descriptors
- org.eclipse.persistence.core.mappings
- org.eclipse.persistence.core.mappings.converters
- org.eclipse.persistence.core.queries
- org.eclipse.persistence.core.sessions
- org.eclipse.persistence.descriptors
- org.eclipse.persistence.descriptors.copying
- org.eclipse.persistence.descriptors.invalidation

- org.eclipse.persistence.descriptors.partitioning
- org.eclipse.persistence.dynamic
- org.eclipse.persistence.eis
- org.eclipse.persistence.eis.interactions
- org.eclipse.persistence.eis.mappings
- org.eclipse.persistence.exceptions
- org.eclipse.persistence.exceptions.i18n
- org.eclipse.persistence.expressions
- org.eclipse.persistence.expressions.spatial
- org.eclipse.persistence.history
- org.eclipse.persistence.internal.cache
- org.eclipse.persistence.internal.codegen
- org.eclipse.persistence.internal.core.databaseaccess
- org.eclipse.persistence.internal.core.descriptors
- org.eclipse.persistence.internal.core.helper
- org.eclipse.persistence.internal.core.queries
- org.eclipse.persistence.internal.core.sessions
- org.eclipse.persistence.internal.databaseaccess
- org.eclipse.persistence.internal.descriptors.changetracking
- org.eclipse.persistence.internal.dynamic
- org.eclipse.persistence.internal.expressions
- org.eclipse.persistence.internal.helper
- org.eclipse.persistence.internal.helper.linkedlist
- org.eclipse.persistence.internal.history
- org.eclipse.persistence.internal.indirection
- org.eclipse.persistence.internal.jpa
- org.eclipse.persistence.internal.jpa.deployment
- org.eclipse.persistence.internal.jpa.deployment.xml.parser
- org.eclipse.persistence.internal.jpa.jdbc
- org.eclipse.persistence.internal.jpa.jpql
- org.eclipse.persistence.internal.jpa.metadata
- org.eclipse.persistence.internal.jpa.metadata.accessors
- org.eclipse.persistence.internal.jpa.metadata.accessors.classes
- org.eclipse.persistence.internal.jpa.metadata.accessors.mappings
- org.eclipse.persistence.internal.jpa.metadata.accessors.objects
- org.eclipse.persistence.internal.jpa.metadata.additionalcriteria
- org.eclipse.persistence.internal.jpa.metadata.cache
- org.eclipse.persistence.internal.jpa.metadata.changetracking
- org.eclipse.persistence.internal.jpa.metadata.columns
- org.eclipse.persistence.internal.jpa.metadata.converters
- org.eclipse.persistence.internal.jpa.metadata.copypolicy
- org.eclipse.persistence.internal.jpa.metadata.inheritance
- org.eclipse.persistence.internal.jpa.metadata.listeners
- org.eclipse.persistence.internal.jpa.metadata.locking
- org.eclipse.persistence.internal.jpa.metadata.mappings

- org.eclipse.persistence.internal.jpa.metadata.multitenant
- org.eclipse.persistence.internal.jpa.metadata.nosql
- org.eclipse.persistence.internal.jpa.metadata.partitioning
- org.eclipse.persistence.internal.jpa.metadata.queries
- org.eclipse.persistence.internal.jpa.metadata.sequencing
- org.eclipse.persistence.internal.jpa.metadata.structures
- org.eclipse.persistence.internal.jpa.metadata.tables
- org.eclipse.persistence.internal.jpa.metadata.transformers
- org.eclipse.persistence.internal.jpa.metadata.xml
- org.eclipse.persistence.internal.jpa.metamodel
- org.eclipse.persistence.internal.jpa.parsing
- org.eclipse.persistence.internal.jpa.parsing.jpql
- org.eclipse.persistence.internal.jpa.parsing.jpql antlr
- org.eclipse.persistence.internal.jpa.querydef
- org.eclipse.persistence.internal.jpa.transaction
- org.eclipse.persistence.internal.jpa.weaving
- org.eclipse.persistence.internal.libraries.antlr.runtime
- org.eclipse.persistence.internal.libraries.antlr.runtime.debug
- org.eclipse.persistence.internal.libraries.antlr.runtime.misc
- org.eclipse.persistence.internal.libraries.antlr.runtime.tree
- org.eclipse.persistence.internal.libraries.asm
- org.eclipse.persistence.internal.libraries.asm.commons
- org.eclipse.persistence.internal.libraries.asm.signature
- org.eclipse.persistence.internal.libraries.asm.tree
- org.eclipse.persistence.internal.libraries.asm.tree.analysis
- org.eclipse.persistence.internal.libraries.asm.util
- org.eclipse.persistence.internal.libraries.asm.xml
- org.eclipse.persistence.internal.localization
- org.eclipse.persistence.internal.localization.i18n
- org.eclipse.persistence.internal.oxm
- org.eclipse.persistence.internal.oxm.accessor
- org.eclipse.persistence.internal.oxm.conversion
- org.eclipse.persistence.internal.oxm.documentpreservation
- org.eclipse.persistence.internal.oxm.mappings
- org.eclipse.persistence.internal.oxm.record
- org.eclipse.persistence.internal.oxm.record.deferred
- org.eclipse.persistence.internal.oxm.record.json
- org.eclipse.persistence.internal.oxm.record.namespaces
- org.eclipse.persistence.internal.oxm.schema
- org.eclipse.persistence.internal.oxm.schema.model
- org.eclipse.persistence.internal.oxm.unmapped
- org.eclipse.persistence.internal.platform.database
- org.eclipse.persistence.internal.queries
- org.eclipse.persistence.internal.security
- org.eclipse.persistence.internal.sequencing

- org.eclipse.persistence.internal.sessions
- org.eclipse.persistence.internal.sessions.coordination
- org.eclipse.persistence.internal.sessions.coordination.broadcast
- org.eclipse.persistence.internal.sessions.coordination.corba
- org.eclipse.persistence.internal.sessions.coordination.corba.sun
- org.eclipse.persistence.internal.sessions.coordination.jms
- org.eclipse.persistence.internal.sessions.coordination.rmi
- org.eclipse.persistence.internal.sessions.coordination.rmi.iiop
- org.eclipse.persistence.internal.sessions.factories
- org.eclipse.persistence.internal.sessions.factories.model
- org.eclipse.persistence.internal.sessions.factories.model.event
- org.eclipse.persistence.internal.sessions.factories.model.log
- org.eclipse.persistence.internal.sessions.factories.model.login
- org.eclipse.persistence.internal.sessions.factories.model.platform
- org.eclipse.persistence.internal.sessions.factories.model.pool
- org.eclipse.persistence.internal.sessions.factories.model.project
- org.eclipse.persistence.internal.sessions.factories.model.property
- org.eclipse.persistence.internal.sessions.factories.model.rcm
- org.eclipse.persistence.internal.sessions.factories.model.rcm.command
- org.eclipse.persistence.internal.sessions.factories.model.sequencing
- org.eclipse.persistence.internal.sessions.factories.model.session
- org.eclipse.persistence.internal.sessions.factories.model.transport
- org.eclipse.persistence.internal.sessions.factories.model.transport.discovery
- org.eclipse.persistence.internal.sessions.factories.model.transport.naming
- org.eclipse.persistence.internal.sessions.remote
- org.eclipse.persistence.jpa.dynamic
- org.eclipse.persistence.jpa.jpql
- org.eclipse.persistence.jpa.jpql.parser
- org.eclipse.persistence.jpa.jpql.tools
- org.eclipse.persistence.jpa.jpql.tools.model
- org.eclipse.persistence.jpa.jpql.tools.model.query
- org.eclipse.persistence.jpa.jpql.tools.resolver
- org.eclipse.persistence.jpa.jpql.tools.spi
- org.eclipse.persistence.jpa.jpql.tools.utility
- org.eclipse.persistence.jpa.jpql.tools.utility.filter
- org.eclipse.persistence.jpa.jpql.tools.utility.iterable
- org.eclipse.persistence.jpa.jpql.tools.utility.iterator
- org.eclipse.persistence.jpa.jpql.utility
- org.eclipse.persistence.jpa.jpql.utility.filter
- org.eclipse.persistence.jpa.jpql.utility.iterable
- org.eclipse.persistence.jpa.jpql.utility.iterator
- org.eclipse.persistence.jpa.metadata
- org.eclipse.persistence.logging
- org.eclipse.persistence.mappings
- org.eclipse.persistence.mappings.converters

- org.eclipse.persistence.mappings.foundation
- org.eclipse.persistence.mappings.querykeys
- org.eclipse.persistence.mappings.structures
- org.eclipse.persistence.mappings.transformers
- org.eclipse.persistence.mappings.xdb
- org.eclipse.persistence.oxm
- org.eclipse.persistence.oxm.annotations
- org.eclipse.persistence.oxm.attachment
- org.eclipse.persistence.oxm.documentpreservation
- org.eclipse.persistence.oxm.mappings
- org.eclipse.persistence.oxm.mappings.converters
- org.eclipse.persistence.oxm.mappings.nullpolicy
- org.eclipse.persistence.oxm.platform
- org.eclipse.persistence.oxm.record
- org.eclipse.persistence.oxm.schema
- org.eclipse.persistence.oxm.sequenced
- org.eclipse.persistence.oxm.unmapped
- org.eclipse.persistence.platform.database
- org.eclipse.persistence.platform.database.converters
- org.eclipse.persistence.platform.database.events
- org.eclipse.persistence.platform.database.jdbc
- org.eclipse.persistence.platform.database.oracle.annotations
- org.eclipse.persistence.platform.database.oracle.jdbc
- org.eclipse.persistence.platform.database.oracle.plsql
- org.eclipse.persistence.platform.database.partitioning
- org.eclipse.persistence.platform.server
- org.eclipse.persistence.platform.xml
- org.eclipse.persistence.platform.xml.jaxp
- org.eclipse.persistence.sequencing
- org.eclipse.persistence.services
- org.eclipse.persistence.services.websphere
- org.eclipse.persistence.sessions.broker
- org.eclipse.persistence.sessions.changesets
- org.eclipse.persistence.sessions.coordination
- org.eclipse.persistence.sessions.coordination.broadcast
- org.eclipse.persistence.sessions.coordination.corba
- org.eclipse.persistence.sessions.coordination.corba.sun
- org.eclipse.persistence.sessions.coordination.jms
- org.eclipse.persistence.sessions.coordination.rmi
- org.eclipse.persistence.sessions.factories
- org.eclipse.persistence.sessions.interceptors
- org.eclipse.persistence.sessions.remote
- org.eclipse.persistence.sessions.remote.corba.sun
- org.eclipse.persistence.sessions.remote.rmi
- org.eclipse.persistence.sessions.remote.rmi.iiop

- org.eclipse.persistence.sessions.serializers
- org.eclipse.persistence.sessions.server
- org.eclipse.persistence.tools
- org.eclipse.persistence.tools.file
- org.eclipse.persistence.tools.profiler
- org.eclipse.persistence.tools.schemaframework
- org.eclipse.persistence.tools.tuning
- org.eclipse.persistence.tools.weaving.jpa
- org.eclipse.persistence.transaction
- org.eclipse.persistence.transaction.was

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Persistence API 2.1 feature:

- classloading
- jpa
- library
- transaction

Java RESTful Services 1.1

This feature enables support for Java API for RESTful Web Services 1.1. JAX-RS annotations can be used to define web service clients and endpoints that comply with the REST architectural style. Endpoints are accessed through a common interface that is based on the HTTP standard methods.

Enabling this feature

To enable the Java RESTful Services 1.1 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jaxrs-1.1</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java RESTful Services 1.1 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jaxrs-1.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- json-1.0 - JavaScript Object Notation for Java
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Standard API packages provided by this feature

- javax.ws.rs
- javax.ws.rs.core
- javax.ws.rs.ext

IBM API packages provided by this feature

- com.ibm.websphere.jaxrs.providers.json4j
- com.ibm.websphere.jaxrs.server

Third party API packages provided by this feature

- org.apache.wink.client
- org.apache.wink.client.handlers
- org.apache.wink.client.internal
- org.apache.wink.client.internal.handlers
- org.apache.wink.client.internal.log
- org.apache.wink.common
- org.apache.wink.common.annotations
- org.apache.wink.common.categories
- org.apache.wink.common.http
- org.apache.wink.common.internal
- org.apache.wink.common.internal.application
- org.apache.wink.common.internal.contexts
- org.apache.wink.common.internal.http
- org.apache.wink.common.internal.i18n
- org.apache.wink.common.internal.lifecycle
- org.apache.wink.common.internal.log
- org.apache.wink.common.internal.model
- org.apache.wink.common.internal.model.admin
- org.apache.wink.common.internal.properties
- org.apache.wink.common.internal.providers.entity
- org.apache.wink.common.internal.providers.entity.app
- org.apache.wink.common.internal.providers.entity.atom
- org.apache.wink.common.internal.providers.entity.csv
- org.apache.wink.common.internal.providers.entity.json
- org.apache.wink.common.internal.providers.entity.xml
- org.apache.wink.common.internal.providers.error
- org.apache.wink.common.internal.providers.header
- org.apache.wink.common.internal.providers.multipart
- org.apache.wink.common.internal.registry
- org.apache.wink.common.internal.registry.metadata
- org.apache.wink.common.internal.runtime
- org.apache.wink.common.internal.uri
- org.apache.wink.common.internal.uritemplate
- org.apache.wink.common.internal.utils
- org.apache.wink.common.model
- org.apache.wink.common.model.app
- org.apache.wink.common.model.atom
- org.apache.wink.common.model.csv
- org.apache.wink.common.model.multipart
- org.apache.wink.common.model.opensearch
- org.apache.wink.common.model.rss
- org.apache.wink.common.model.synd
- org.apache.wink.common.model.wadl
- org.apache.wink.common.utils

- org.apache.wink.jcdi.server.internal
- org.apache.wink.jcdi.server.internal.extension
- org.apache.wink.jcdi.server.internal.lifecycle
- org.apache.wink.providers.abdera
- org.apache.wink.providers.jackson
- org.apache.wink.providers.json4j
- org.apache.wink.server.handlers
- org.apache.wink.server.internal
- org.apache.wink.server.internal.application
- org.apache.wink.server.internal.contexts
- org.apache.wink.server.internal.handlers
- org.apache.wink.server.internal.handlers.ejb
- org.apache.wink.server.internal.lifecycle
- org.apache.wink.server.internal.lifecycle.metadata
- org.apache.wink.server.internal.log
- org.apache.wink.server.internal.providers.entity.html
- org.apache.wink.server.internal.providers.exception
- org.apache.wink.server.internal.registry
- org.apache.wink.server.internal.resources
- org.apache.wink.server.internal.servlet
- org.apache.wink.server.internal.servlet.contentencode
- org.apache.wink.server.internal.utils
- org.apache.wink.server.utils
- org.codehaus.jackson
- org.codehaus.jackson.annotate
- org.codehaus.jackson.impl
- org.codehaus.jackson.io
- org.codehaus.jackson.jaxrs
- org.codehaus.jackson.map
- org.codehaus.jackson.map.annotate
- org.codehaus.jackson.map.deser
- org.codehaus.jackson.map.exc
- org.codehaus.jackson.map.introspect
- org.codehaus.jackson.map.jsontype
- org.codehaus.jackson.map.jsontype.impl
- org.codehaus.jackson.map.ser
- org.codehaus.jackson.map.type
- org.codehaus.jackson.map.util
- org.codehaus.jackson.node
- org.codehaus.jackson.schema
- org.codehaus.jackson.sym
- org.codehaus.jackson.type
- org.codehaus.jackson.util
- org.codehaus.jackson.xc

Java RESTful Services 2.0

This feature enables support for Java API for RESTful Web Services. JAX-RS annotations can be used to define web service clients and endpoints that comply with the REST architectural style. Endpoints are accessed through a common interface that is based on the HTTP standard methods.

Enabling this feature

To enable the Java RESTful Services 2.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jaxrs-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java RESTful Services 2.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jaxrs-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `jaxrsClient-2.0` - Java RESTful Services Client 2.0
- `json-1.0` - JavaScript Object Notation for Java
- `servlet-3.1` - Java Servlets 3.1

Features that enable this feature

- `microProfile-1.0` - Micro Profile 1.0
- `webProfile-7.0` - Java EE Web Profile 7.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java RESTful Services 2.0 feature:

- `classloading`
- `library`

Java RESTful Services Client 2.0

This feature enables support for Java Client API for JAX-RS 2.0.

Enabling this feature

To enable the Java RESTful Services Client 2.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jaxrsClient-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java RESTful Services Client 2.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jaxrsClient-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `json-1.0` - JavaScript Object Notation for Java
- `servlet-3.1` - Java Servlets 3.1

Features that enable this feature

- jaxrs-2.0 - Java RESTful Services 2.0

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java RESTful Services Client 2.0 feature:

- classloading
- library

Java Servlets 3.0

This feature enables support for HTTP Servlets written to the Java Servlet 3.0 specification. The servlets can be packaged in Java EE specified WAR or EAR files. If servlet security is required, an `appSecurity` feature should also be configured; in the absence of a security feature any security constraints for the application will be ignored.

Enabling this feature

To enable the Java Servlets 3.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>servlet-3.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Servlets 3.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.servlet-3.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- adminCenter-1.0 - Admin Center
- apiDiscovery-1.0 - API Discovery 1.0
- appSecurity-1.0 - Application Security 1.0
- collectiveMember-1.0 - Collective Member
- httpWhiteboard-1.0 - OSGi Http Whiteboard
- jaspic-1.1 - Java Authentication SPI for Containers 1.1
- jaxrs-1.1 - Java RESTful Services 1.1
- jpa-2.0 - Java Persistence API 2.0
- jsf-2.0 - JavaServer Faces 2.0
- jsp-2.2 - JavaServer Pages 2.2
- oauth-2.0 - OAuth
- openid-2.0 - OpenID
- openidConnectClient-1.0 - OpenID Connect Client
- openidConnectServer-1.0 - OpenID Connect Provider
- osgi.jpa-1.0 - OSGi Java Persistence API
- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- samlWeb-2.0 - SAML web single sign-on version 2.0
- scim-1.0 - System for Cross-domain Identity Management
- spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism
- wab-1.0 - OSGi Web Application Bundles

- webCache-1.0 - Web Response Cache
- webProfile-6.0 - Java EE Web Profile 6.0

Standard API packages provided by this feature

- javax.servlet
- javax.servlet.annotation
- javax.servlet.descriptor
- javax.servlet.http
- javax.servlet.resources

IBM API packages provided by this feature

- com.ibm.websphere.servlet.container
- com.ibm.websphere.servlet.context
- com.ibm.websphere.servlet.error
- com.ibm.websphere.servlet.event
- com.ibm.websphere.servlet.session
- com.ibm.websphere.webcontainer
- com.ibm.wsspi.servlet.session

SPI packages provided by this feature

- com.ibm.websphere.servlet.filter
- com.ibm.websphere.servlet.request
- com.ibm.websphere.servlet.response
- com.ibm.websphere.webcontainer.async
- com.ibm.ws.webcontainer.extension
- com.ibm.ws.webcontainer.spiadapter.collaborator
- com.ibm.wsspi.webcontainer
- com.ibm.wsspi.webcontainer.collaborator
- com.ibm.wsspi.webcontainer.extension
- com.ibm.wsspi.webcontainer.filter
- com.ibm.wsspi.webcontainer.metadata
- com.ibm.wsspi.webcontainer.osgi.extension
- com.ibm.wsspi.webcontainer.servlet
- com.ibm.wsspi.webcontainer.webapp

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Servlets 3.0 feature:

- application
- applicationManager
- applicationMonitor
- channelfw
- classloading
- cors
- enterpriseApplication
- httpAccessLogging
- httpDispatcher

- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- httpSession
- javaPermission
- library
- mimeTypeypes
- pluginConfiguration
- tcpOptions
- virtualHost
- webApplication
- webContainer

Java Servlets 3.1

This feature enables support for HTTP Servlets written to the Java Servlet 3.1 specification. You can package servlets in Java EE specified WAR or EAR files. If servlet security is required, you should also configure an appSecurity feature. Without a security feature, any security constraints for the application are ignored.

Enabling this feature

To enable the Java Servlets 3.1 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>servlet-3.1</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java Servlets 3.1 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.servlet-3.1; type="osgi.subsystem.feature"
```

Features that enable this feature

- adminCenter-1.0 - Admin Center
- apiDiscovery-1.0 - API Discovery 1.0
- appSecurity-1.0 - Application Security 1.0
- collectiveMember-1.0 - Collective Member
- httpWhiteboard-1.0 - OSGi Http Whiteboard
- jaspic-1.1 - Java Authentication SPI for Containers 1.1
- jaxrs-1.1 - Java RESTful Services 1.1
- jaxrs-2.0 - Java RESTful Services 2.0
- jaxrsClient-2.0 - Java RESTful Services Client 2.0
- jpa-2.0 - Java Persistence API 2.0
- jsf-2.0 - JavaServer Faces 2.0
- jsf-2.2 - JavaServer Faces 2.2
- jsp-2.2 - JavaServer Pages 2.2
- jsp-2.3 - JavaServer Pages 2.3
- oauth-2.0 - OAuth
- openid-2.0 - OpenID

- openidConnectClient-1.0 - OpenID Connect Client
- openidConnectServer-1.0 - OpenID Connect Provider
- osgi.jpaa-1.0 - OSGi Java Persistence API
- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- samlWeb-2.0 - SAML web single sign-on version 2.0
- scim-1.0 - System for Cross-domain Identity Management
- spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism
- wab-1.0 - OSGi Web Application Bundles
- webCache-1.0 - Web Response Cache
- webProfile-7.0 - Java EE Web Profile 7.0
- websocket-1.0 - Java WebSocket 1.0
- websocket-1.1 - Java WebSocket 1.1

Standard API packages provided by this feature

- javax.servlet
- javax.servlet.annotation
- javax.servlet.descriptor
- javax.servlet.http
- javax.servlet.resources

IBM API packages provided by this feature

- com.ibm.websphere.servlet.container
- com.ibm.websphere.servlet.context
- com.ibm.websphere.servlet.error
- com.ibm.websphere.servlet.event
- com.ibm.websphere.servlet.session
- com.ibm.websphere.webcontainer
- com.ibm.wsspi.servlet.session

SPI packages provided by this feature

- com.ibm.websphere.servlet.filter
- com.ibm.websphere.servlet.request
- com.ibm.websphere.servlet.response
- com.ibm.websphere.webcontainer.async
- com.ibm.ws.webcontainer.extension
- com.ibm.ws.webcontainer.spiadapter.collaborator
- com.ibm.wsspi.webcontainer
- com.ibm.wsspi.webcontainer.collaborator
- com.ibm.wsspi.webcontainer.extension
- com.ibm.wsspi.webcontainer.filter
- com.ibm.wsspi.webcontainer.metadata
- com.ibm.wsspi.webcontainer.osgi.extension
- com.ibm.wsspi.webcontainer.servlet
- com.ibm.wsspi.webcontainer.webapp

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java Servlets 3.1 feature:

- `application`
- `applicationManager`
- `applicationMonitor`
- `channelfw`
- `classloading`
- `cors`
- `enterpriseApplication`
- `httpAccessLogging`
- `httpDispatcher`
- `httpEncoding`
- `httpEndpoint`
- `httpOptions`
- `httpProxyRedirect`
- `httpSession`
- `javaPermission`
- `library`
- `mimeTypes`
- `pluginConfiguration`
- `tcpOptions`
- `virtualHost`
- `webApplication`
- `webContainer`

Java WebSocket 1.0

This feature enables support for WebSocket applications written to the Java API for WebSocket 1.0 specification.

Enabling this feature

To enable the Java WebSocket 1.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>websocket-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java WebSocket 1.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.websocket-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `servlet-3.1` - Java Servlets 3.1

Standard API packages provided by this feature

- `javax.websocket`
- `javax.websocket.server`

IBM API packages provided by this feature

- com.ibm.websphere.wsoc

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java WebSocket 1.0 feature:

- wsocOutbound

Java WebSocket 1.1

This feature enables support for WebSocket applications written to the Java API for WebSocket 1.1 specification.

Enabling this feature

To enable the Java WebSocket 1.1 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>websocket-1.1</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Java WebSocket 1.1 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.websocket-1.1; type="osgi.subsystem.feature"
```

Features that this feature enables

- servlet-3.1 - Java Servlets 3.1

Features that enable this feature

- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.websocket
- javax.websocket.server

IBM API packages provided by this feature

- com.ibm.websphere.wsoc

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Java WebSocket 1.1 feature:

- wsocOutbound

JavaMail 1.5

This feature allows applications to utilize the JavaMail 1.5 API.

Enabling this feature

To enable the JavaMail 1.5 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>javaMail-1.5</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaMail 1.5 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.javaMail-1.5; type="osgi.subsystem.feature"
```

Standard API packages provided by this feature

- javax.mail
- javax.mail.event
- javax.mail.internet
- javax.mail.search
- javax.mail.util

Third party API packages provided by this feature

- com.sun.mail
- com.sun.mail.auth
- com.sun.mail.handlers
- com.sun.mail.iap
- com.sun.mail.imap
- com.sun.mail.imap.protocol
- com.sun.mail.pop3
- com.sun.mail.smtp
- com.sun.mail.util
- com.sun.mail.util.logging

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the JavaMail 1.5 feature:

- `mailSession`

JavaScript Object Notation Processing

The Java API for JSON Processing (JSON-P) feature provides a standardized method for constructing and manipulating data to be rendered in JavaScript Object Notation (JSON).

Enabling this feature

To enable the JavaScript Object Notation Processing feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jsonp-1.0</feature>
```

Supported Java Versions

- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaScript Object Notation Processing feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jsonp-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- microProfile-1.0 - Micro Profile 1.0
- webProfile-7.0 - Java EE Web Profile 7.0

Standard API packages provided by this feature

- javax.json
- javax.json.spi
- javax.json.stream

JavaScript Object Notation for Java

This feature provides access to the JavaScript Object Notation (JSON4J) library. The JSON4J library provides a simple Java model for constructing and manipulating data to be rendered as JSON data.

Enabling this feature

To enable the JavaScript Object Notation for Java feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jsonp-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaScript Object Notation for Java feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.json-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- adminCenter-1.0 - Admin Center
- apiDiscovery-1.0 - API Discovery 1.0
- jaxrs-1.1 - Java RESTful Services 1.1
- jaxrs-2.0 - Java RESTful Services 2.0
- jaxrsClient-2.0 - Java RESTful Services Client 2.0
- oauth-2.0 - OAuth
- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- scim-1.0 - System for Cross-domain Identity Management

IBM API packages provided by this feature

- com.ibm.json.java
- com.ibm.json.xml

JavaServer Faces 2.0

This feature enables support for web applications that use the Java Server Faces (JSF) framework. This framework simplifies the construction of user interfaces.

Enabling this feature

To enable the JavaServer Faces 2.0 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jsf-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaServer Faces 2.0 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jsf-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `beanValidation-1.0` - Bean Validation 1.0
- `beanValidation-1.1` - Bean Validation 1.1
- `jsp-2.2` - JavaServer Pages 2.2
- `jsp-2.3` - JavaServer Pages 2.3
- `servlet-3.0` - Java Servlets 3.0
- `servlet-3.1` - Java Servlets 3.1

Features that enable this feature

- `webProfile-6.0` - Java EE Web Profile 6.0

Standard API packages provided by this feature

- `javax.faces`
- `javax.faces.application`
- `javax.faces.bean`
- `javax.faces.component`
- `javax.faces.component.behavior`
- `javax.faces.component.html`
- `javax.faces.component.visit`
- `javax.faces.context`
- `javax.faces.convert`
- `javax.faces.el`
- `javax.faces.event`
- `javax.faces.lifecycle`
- `javax.faces.model`
- `javax.faces.render`
- `javax.faces.validator`
- `javax.faces.view`
- `javax.faces.view.facelets`
- `javax.faces.webapp`
- `javax.persistence`

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the JavaServer Faces 2.0 feature:

- `classloading`
- `transaction`

JavaServer Faces 2.2

This feature enables support for web applications that use the Java Server Faces (JSF) 2.2 framework. This framework simplifies the construction of user interfaces.

Enabling this feature

To enable the JavaServer Faces 2.2 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jsf-2.2</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaServer Faces 2.2 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jsf-2.2; type="osgi.subsystem.feature"
```

Features that this feature enables

- `jsp-2.3` - JavaServer Pages 2.3
- `servlet-3.1` - Java Servlets 3.1

Features that enable this feature

- `webProfile-7.0` - Java EE Web Profile 7.0

Standard API packages provided by this feature

- `javax.faces`
- `javax.faces.application`
- `javax.faces.bean`
- `javax.faces.component`
- `javax.faces.component.behavior`
- `javax.faces.component.html`
- `javax.faces.component.visit`
- `javax.faces.context`
- `javax.faces.convert`
- `javax.faces.el`
- `javax.faces.event`
- `javax.faces.flow`
- `javax.faces.flow.builder`
- `javax.faces.lifecycle`
- `javax.faces.model`
- `javax.faces.render`
- `javax.faces.validator`
- `javax.faces.view`
- `javax.faces.view.facelets`

- javax.faces.webapp

JavaServer Pages 2.2

This feature enables support for Java Server Pages (JSPs) that are written to the JSP 2.2 specification. This framework simplifies the construction of user interfaces.

Enabling this feature

To enable the JavaServer Pages 2.2 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>jsp-2.2</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaServer Pages 2.2 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jsp-2.2; type="osgi.subsystem.feature"
```

Features that this feature enables

- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Features that enable this feature

- adminCenter-1.0 - Admin Center
- jsf-2.0 - JavaServer Faces 2.0
- oauth-2.0 - OAuth
- openidConnectClient-1.0 - OpenID Connect Client
- openidConnectServer-1.0 - OpenID Connect Provider
- webCache-1.0 - Web Response Cache
- webProfile-6.0 - Java EE Web Profile 6.0

Standard API packages provided by this feature

- javax.el
- javax.servlet.jsp
- javax.servlet.jsp.el
- javax.servlet.jsp.jstl.core
- javax.servlet.jsp.jstl.fmt
- javax.servlet.jsp.jstl.sql
- javax.servlet.jsp.jstl.tlv
- javax.servlet.jsp.resources
- javax.servlet.jsp.tagext

SPI packages provided by this feature

- com.ibm.wsspi.jsp.taglib.config

Feature configuration elements

You can use the following elements in your server.xml file to configure the JavaServer Pages 2.2 feature:

- jspEngine

JavaServer Pages 2.3

This feature enables support for Java Server Pages (JSPs) that are written to the JSP 2.3 specification. This framework simplifies the construction of user interfaces. Enabling this feature also enables the Expression Language (EL) version 3.0 feature.

Enabling this feature

To enable the JavaServer Pages 2.3 feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>jsp-2.3</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the JavaServer Pages 2.3 feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.jsp-2.3; type="osgi.subsystem.feature"
```

Features that this feature enables

- `el-3.0` - Expression Language 3.0
- `servlet-3.1` - Java Servlets 3.1

Features that enable this feature

- `adminCenter-1.0` - Admin Center
- `jsf-2.0` - JavaServer Faces 2.0
- `jsf-2.2` - JavaServer Faces 2.2
- `oauth-2.0` - OAuth
- `openidConnectClient-1.0` - OpenID Connect Client
- `openidConnectServer-1.0` - OpenID Connect Provider
- `webCache-1.0` - Web Response Cache
- `webProfile-7.0` - Java EE Web Profile 7.0

Standard API packages provided by this feature

- `javax.el`
- `javax.servlet.jsp`
- `javax.servlet.jsp.el`
- `javax.servlet.jsp.jstl.core`
- `javax.servlet.jsp.jstl.fmt`
- `javax.servlet.jsp.jstl.sql`
- `javax.servlet.jsp.jstl.tlv`
- `javax.servlet.jsp.tagext`

SPI packages provided by this feature

- `com.ibm.wsspi.jsp.taglib.config`

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the JavaServer Pages 2.3 feature:

- `jspEngine`

Job Manager Integration

This feature enables Liberty profile servers to automatically publish their status to the WebSphere Application Server Job Managers. This feature is required for the Job Manager to discover server instances that it did not start.

Enabling this feature

To enable the Job Manager Integration feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>serverStatus-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Job Manager Integration feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.serverStatus-1.0; type="osgi.subsystem.feature"
```

Kerberos Constrained Delegation for SPNEGO

This feature enables support for Kerberos constrained delegation for SPNEGO.

Enabling this feature

To enable the Kerberos Constrained Delegation for SPNEGO feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>constrainedDelegation-1.0</feature>
```

Supported Java Versions

- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Kerberos Constrained Delegation for SPNEGO feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.constrainedDelegation-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `appSecurity-2.0` - Application Security 2.0

IBM API packages provided by this feature

- `com.ibm.websphere.security.s4u2proxy`

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Kerberos Constrained Delegation for SPNEGO feature:

- `constrainedDelegation`

LDAP User Registry

This feature enables support for using an LDAP server as a user registry. Any server that supports LDAP Version 3.0 may be used. Multiple LDAP registries can be configured, and then federated to achieve a single logical registry view.

Enabling this feature

To enable the LDAP User Registry feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>ldapRegistry-3.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the LDAP User Registry feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.ldapRegistry-3.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `federatedRegistry-1.0` - Federated User Registry

Features that enable this feature

- `appSecurity-1.0` - Application Security 1.0
- `oauth-2.0` - OAuth

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the LDAP User Registry feature:

- `activedLdapFilterProperties`
- `administrator-role`
- `classloading`
- `customLdapFilterProperties`
- `domino50LdapFilterProperties`
- `edirectoryLdapFilterProperties`
- `idsLdapFilterProperties`
- `iplanetLdapFilterProperties`
- `ldapRegistry`
- `library`
- `netscapeLdapFilterProperties`
- `securewayLdapFilterProperties`

Logstash Collector 1.0

Logstash collector gathers data from various sources and forwards the data to a Logstash server using Lumberjack protocol.

Enabling this feature

To enable the Logstash Collector 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>logstashCollector-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Logstash Collector 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.logstashCollector-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- ssl-1.0 - Secure Socket Layer

Feature configuration elements

You can use the following elements in your server.xml file to configure the Logstash Collector 1.0 feature:

- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- logstashCollector
- mimeTypeypes
- tcpOptions
- virtualHost

Micro Profile 1.0

This feature combines the Liberty features that support the Micro Profile for enterprise Java.

Enabling this feature

To enable the Micro Profile 1.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>microProfile-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Micro Profile 1.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.microProfile-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- cdi-1.2 - Contexts and Dependency Injection 1.2
- jaxrs-2.0 - Java RESTful Services 2.0
- jsonp-1.0 - JavaScript Object Notation Processing

OAuth

This feature enables web applications to integrate OAuth 2.0 for authenticating and authorizing users.

Enabling this feature

To enable the OAuth feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>oauth-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OAuth feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.oauth-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- json-1.0 - JavaScript Object Notation for Java
- jsp-2.2 - JavaServer Pages 2.2
- jsp-2.3 - JavaServer Pages 2.3
- ldapRegistry-3.0 - LDAP User Registry
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

Features that enable this feature

- openidConnectClient-1.0 - OpenID Connect Client
- openidConnectServer-1.0 - OpenID Connect Provider

IBM API packages provided by this feature

- com.ibm.oauth.core.api.attributes
- com.ibm.oauth.core.api.config
- com.ibm.oauth.core.api.error
- com.ibm.oauth.core.api.error.oauth20
- com.ibm.oauth.core.api.oauth20.mediator
- com.ibm.websphere.security.oauth20
- com.ibm.websphere.security.openidconnect.token
- com.ibm.wsspi.security.oauth20.token

SPI packages provided by this feature

- com.ibm.wsspi.security.oauth20
- com.ibm.wsspi.security.openidconnect

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OAuth feature:

- administrator-role
- authCache
- authentication
- authorization-roles
- basicRegistry
- classloading
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- oauth-roles
- oauthProvider
- quickStartSecurity
- trustAssociation

OSGi Application Integration

This feature adds local application-to-application integration for OSGi Applications.

Enabling this feature

To enable the OSGi Application Integration feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>osgiAppIntegration-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Application Integration feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.ws.eba.app.integration-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- blueprint-1.0 - OSGi Blueprint

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Application Integration feature:

- classloading
- transaction

OSGi Blueprint

This feature enables support for deploying OSGi applications that use the OSGi blueprint container specification. With the OSGi Applications support in WebSphere Application Server, you can develop and deploy modular applications that use Java EE and OSGi technologies.

Enabling this feature

To enable the OSGi Blueprint feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>blueprint-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Blueprint feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.blueprint-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- osgi.jpaa-1.0 - OSGi Java Persistence API
- osgiAppIntegration-1.0 - OSGi Application Integration
- wab-1.0 - OSGi Web Application Bundles

Standard API packages provided by this feature

- org.apache.aries.blueprint
- org.apache.aries.blueprint.ext
- org.apache.aries.blueprint.ext.evaluator
- org.apache.aries.blueprint.mutable
- org.apache.aries.blueprint.services
- org.apache.aries.blueprint.utils
- org.apache.aries.util
- org.osgi.framework
- org.osgi.framework.hooks.bundle
- org.osgi.framework.hooks.resolver
- org.osgi.framework.hooks.service
- org.osgi.framework.hooks.weaving
- org.osgi.framework.launch
- org.osgi.framework.namespace
- org.osgi.framework.startlevel
- org.osgi.framework.wiring
- org.osgi.resource
- org.osgi.service.blueprint
- org.osgi.service.blueprint.container
- org.osgi.service.blueprint.reflect
- org.osgi.service.component

- org.osgi.service.condpermadmin
- org.osgi.service.jndi
- org.osgi.service.packageadmin
- org.osgi.service.startlevel
- org.osgi.service.url
- org.osgi.util.tracker

Third party API packages provided by this feature

- org.apache.aries.transaction.exception

SPI packages provided by this feature

- org.apache.aries.blueprint
- org.apache.aries.blueprint.ext
- org.apache.aries.blueprint.ext.evaluator
- org.apache.aries.blueprint.mutable
- org.apache.aries.blueprint.services
- org.apache.aries.util
- org.osgi.service.cm
- org.osgi.service.subsystem

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Blueprint feature:

- application
- applicationManager
- applicationMonitor
- bundleRepository
- classloading
- javaPermission
- library
- osgiApplication
- osgiApplications
- osgiLibrary
- transaction

OSGi Bundle

This feature enables support for deploying OSGi applications. With the OSGi Applications support in WebSphere Application Server, you can develop and deploy modular applications that use Java EE and OSGi technologies.

Enabling this feature

To enable the OSGi Bundle feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>osgiBundle-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7

- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Bundle feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.osgiBundle-1.0; type="osgi.subsystem.feature"
```

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Bundle feature:

- application
- applicationManager
- applicationMonitor
- bundleRepository
- classloading
- javaPermission
- library
- osgiApplication
- osgiApplications
- osgiLibrary

OSGi Debug Console

This feature enables an OSGi console to aid debug of the runtime. It can be used to display information about bundles, packages and services which may be useful when developing your own features for product extensions.

Enabling this feature

To enable the OSGi Debug Console feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>osgiConsole-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Debug Console feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.osgiConsole-1.0; type="osgi.subsystem.feature"
```

OSGi Http Whiteboard

This feature enables deploying modular Web applications written using Servlet technologies and the OSGi HTTP Whiteboard specification.

Enabling this feature

To enable the OSGi Http Whiteboard feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>httpWhiteboard-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Http Whiteboard feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.httpWhiteboard-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Standard API packages provided by this feature

- org.osgi.service.http
- org.osgi.service.http.context
- org.osgi.service.http.whiteboard

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Http Whiteboard feature:

- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- httpWhiteboard
- mimeTypeypes
- tcpOptions
- virtualHost

OSGi Java Persistence API

This feature is superseded by the `blueprint-1.0` and `jpa-2.0` features. When those features are both added to the server, this feature is added automatically.

Enabling this feature

To enable the OSGi Java Persistence API feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>osgi.jpa-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Java Persistence API feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.osgi.jpa-1.0; type="osgi.subsystem.feature"
```

Features that this feature is superseded by

- blueprint-1.0 - OSGi Blueprint
- jpa-2.0 - Java Persistence API 2.0

Features that this feature enables

- beanValidation-1.0 - Bean Validation 1.0
- beanValidation-1.1 - Bean Validation 1.1

- blueprint-1.0 - OSGi Blueprint
- jdbc-4.0 - Java Database Connectivity 4.0
- jdbc-4.1 - Java Database Connectivity 4.1
- jndi-1.0 - Java Naming and Directory Interface
- jpa-2.0 - Java Persistence API 2.0
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Java Persistence API feature:

- classloading
- transaction

OSGi Web Application Bundles

This feature enables OSGi applications that contain Web Application Bundles (WABs). Web application bundles are OSGi bundles that are internally structured in the same way as a war file and support the same web components.

Enabling this feature

To enable the OSGi Web Application Bundles feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>wab-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OSGi Web Application Bundles feature, include the following item in the `Subsystem-Content` header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.wab-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- blueprint-1.0 - OSGi Blueprint
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OSGi Web Application Bundles feature:

- classloading
- transaction

OpenID

This feature enables web applications to integrate OpenID 2.0 for authenticating users instead of, or in addition to, the configured user registry.

Enabling this feature

To enable the OpenID feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>openid-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OpenID feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.openid-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OpenID feature:

- `authFilter`
- `openId`
- `userInfo`

OpenID Connect Client

This feature enables web applications to integrate OpenID Connect Client 1.0 for authenticating users instead of, or in addition to, the configured user registry.

Enabling this feature

To enable the OpenID Connect Client feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>openidConnectClient-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OpenID Connect Client feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.openidConnectClient-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- jsp-2.2 - JavaServer Pages 2.2
- jsp-2.3 - JavaServer Pages 2.3
- oauth-2.0 - OAuth
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

IBM API packages provided by this feature

- `com.ibm.websphere.security.openidconnect`

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the OpenID Connect Client feature:

- administrator-role
- authCache
- authFilter
- authentication
- authorization-roles
- basicRegistry
- classloading
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- openidConnectClient
- quickStartSecurity
- trustAssociation

OpenID Connect Provider

This feature enables web applications to integrate OpenID Connect Server 1.0 for authenticating users instead of, or in addition to, the configured user registry.

Enabling this feature

To enable the OpenID Connect Provider feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>openidConnectServer-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the OpenID Connect Provider feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.openidConnectServer-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- jsp-2.2 - JavaServer Pages 2.2
- jsp-2.3 - JavaServer Pages 2.3
- oauth-2.0 - OAuth
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

IBM API packages provided by this feature

- com.ibm.websphere.security.openidconnect

Feature configuration elements

You can use the following elements in your server.xml file to configure the OpenID Connect Provider feature:

- openidConnectProvider

Performance Monitoring

This feature enables the Performance Monitoring Infrastructure (PMI) for other features the server is running. Monitoring data is accessible through standard MXBeans; access through the traditional WebSphere Perf MBean can be enabled.

Enabling this feature

To enable the Performance Monitoring feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>monitor-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Performance Monitoring feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.monitor-1.0; type="osgi.subsystem.feature"
```

IBM API packages provided by this feature

- com.ibm.websphere.monitor.jmx

Feature configuration elements

You can use the following elements in your server.xml file to configure the Performance Monitoring feature:

- classloading
- monitor

Request Timing

Provides warnings and diagnostic info for slow or hung requests.

Enabling this feature

To enable the Request Timing feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>requestTiming-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Request Timing feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.requestTiming-1.0; type="osgi.subsystem.feature"
```

Feature configuration elements

You can use the following elements in your server.xml file to configure the Request Timing feature:

- requestTiming

SAML web single sign-on version 2.0

This feature enables web applications to use SAML web single sign-on version 2.0 function.

Enabling this feature

To enable the SAML web single sign-on version 2.0 feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>samlWeb-2.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the SAML web single sign-on version 2.0 feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.samlWeb-2.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1
- ssl-1.0 - Secure Socket Layer

Feature configuration elements

You can use the following elements in your server.xml file to configure the SAML web single sign-on version 2.0 feature:

- administrator-role
- authCache
- authFilter
- authentication
- authorization-roles
- basicRegistry
- classloading
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- quickStartSecurity
- samlWebSso20
- trustAssociation

Secure Socket Layer

This feature enables support for Secure Sockets Layer (SSL) connections. The secure HTTPS listener is not started unless the ssl-1.0 feature is enabled and a keystore is configured.

Enabling this feature

To enable the Secure Socket Layer feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>ssl-1.0</feature>
```


Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Secure Socket Layer feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.ssl-1.0; type="osgi.subsystem.feature"
```

Features that enable this feature

- adminCenter-1.0 - Admin Center
- apiDiscovery-1.0 - API Discovery 1.0
- appSecurity-2.0 - Application Security 2.0
- federatedRegistry-1.0 - Federated User Registry
- logstashCollector-1.0 - Logstash Collector 1.0
- oauth-2.0 - OAuth
- openidConnectClient-1.0 - OpenID Connect Client
- restConnector-1.0 - JMX REST Connector 1.0
- restConnector-2.0 - JMX REST Connector 2.0
- samlWeb-2.0 - SAML web single sign-on version 2.0
- scim-1.0 - System for Cross-domain Identity Management

IBM API packages provided by this feature

- com.ibm.websphere.ssl

SPI packages provided by this feature

- com.ibm.wsspi.ssl

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Secure Socket Layer feature:

- channelfw
- keyStore
- ssl
- sslDefault
- sslOptions
- tcpOptions

Simple and Protected GSSAPI Negotiation Mechanism

This feature enables web applications to integrate SPNEGO 1.0 for authenticating users instead of, or in addition to, the configured user registry.

Enabling this feature

To enable the Simple and Protected GSSAPI Negotiation Mechanism feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>spnego-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Simple and Protected GSSAPI Negotiation Mechanism feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.spnego-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- appSecurity-2.0 - Application Security 2.0
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Simple and Protected GSSAPI Negotiation Mechanism feature:

- `authFilter`
- `spnego`

System for Cross-domain Identity Management

This feature enables support for invoking User Management API using SCIM REST Services.

Enabling this feature

To enable the System for Cross-domain Identity Management feature, add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>scim-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the System for Cross-domain Identity Management feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.scim-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- `distributedMap-1.0` - Distributed Map interface for Dynamic Caching
- `federatedRegistry-1.0` - Federated User Registry
- `json-1.0` - JavaScript Object Notation for Java
- `servlet-3.0` - Java Servlets 3.0
- `servlet-3.1` - Java Servlets 3.1
- `ssl-1.0` - Secure Socket Layer

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the System for Cross-domain Identity Management feature:

- `administrator-role`
- `authCache`
- `authentication`
- `authorization-roles`
- `basicRegistry`

- channelfw
- classloading
- httpAccessLogging
- httpDispatcher
- httpEncoding
- httpEndpoint
- httpOptions
- httpProxyRedirect
- jaasLoginContextEntry
- jaasLoginModule
- library
- ltpa
- mimeTypees
- quickStartSecurity
- tcpOptions
- trustAssociation
- virtualHost

Timed Operations

This feature enables support for logging warnings when certain operations in the application server are running more slowly than expected.

Enabling this feature

To enable the Timed Operations feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>timedOperations-1.0</feature>
```

Supported Java Versions

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

Developing a feature that depends on this feature

If you are developing a feature that depends on the Timed Operations feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.timedOperations-1.0; type="osgi.subsystem.feature"
```

SPI packages provided by this feature

- com.ibm.wsspi.timedoperations

Feature configuration elements

You can use the following elements in your server.xml file to configure the Timed Operations feature:

- timedOperation

Web Response Cache

This feature enables local caching for web responses. It includes the distributedMap feature and performs automatic caching of web application responses to improve response times and throughput. Applications

can include a cache-spec.xml file to customize the response caching. The cache may be distributed through addition of a network cache provider such as WebSphere eXtreme Scale.

Enabling this feature

To enable the Web Response Cache feature, add the following element declaration inside the featureManager element in your server.xml file:

```
<feature>webCache-1.0</feature>
```

Developing a feature that depends on this feature

If you are developing a feature that depends on the Web Response Cache feature, include the following item in the Subsystem-Content header in the feature manifest file for your new feature:

```
com.ibm.websphere.appserver.webCache-1.0; type="osgi.subsystem.feature"
```

Features that this feature enables

- distributedMap-1.0 - Distributed Map interface for Dynamic Caching
- jsp-2.2 - JavaServer Pages 2.2
- jsp-2.3 - JavaServer Pages 2.3
- servlet-3.0 - Java Servlets 3.0
- servlet-3.1 - Java Servlets 3.1

IBM API packages provided by this feature

- com.ibm.websphere.command
- com.ibm.websphere.command.web
- com.ibm.websphere.servlet.cache

SPI packages provided by this feature

- com.ibm.wsspi.cache.web

Liberty Kernel

The kernel

IBM API packages provided by this feature

- com.ibm.websphere.config.mbeans
- com.ibm.websphere.logging
- com.ibm.websphere.logging.hpel
- com.ibm.websphere.logging.hpel.reader
- com.ibm.websphere.logging.hpel.reader.filters
- com.ibm.websphere.logging.hpel.writer

SPI packages provided by this feature

- com.ibm.websphere.crypto
- com.ibm.websphere.ras
- com.ibm.websphere.ras
- com.ibm.websphere.ras.annotation
- com.ibm.websphere.ras.annotation
- com.ibm.ws.ffdc
- com.ibm.ws.ffdc

- com.ibm.wsspi.config
- com.ibm.wsspi.kernel.filemonitor
- com.ibm.wsspi.kernel.service.location
- com.ibm.wsspi.kernel.service.utils
- com.ibm.wsspi.logging
- com.ibm.wsspi.logging
- com.ibm.wsspi.security.crypto
- com.ibm.wsspi.threading
- org.eclipse.equinox.log
- org.eclipse.osgi.framework.console
- org.eclipse.osgi.framework.eventmgr
- org.eclipse.osgi.framework.log
- org.eclipse.osgi.service.datalocation
- org.eclipse.osgi.service.debug
- org.eclipse.osgi.service.environment
- org.eclipse.osgi.service.localization
- org.eclipse.osgi.service.resolver
- org.eclipse.osgi.service.runnable
- org.eclipse.osgi.service.security
- org.eclipse.osgi.service.urlconversion
- org.eclipse.osgi.signedcontent
- org.eclipse.osgi.storagemanager
- org.eclipse.osgi.util
- org.osgi.service.cm
- org.osgi.service.component
- org.osgi.service.coordinator
- org.osgi.service.event
- org.osgi.service.log
- org.osgi.service.metatype

Feature configuration elements

You can use the following elements in your `server.xml` file to configure the Liberty Kernel feature:

- config
- executor
- featureManager
- fileset
- include
- logging
- variable
- zosLogging


Liberty Repository



8.5.5.2

The Liberty Repository provides an online mechanism to deliver Liberty and additional content, enabling a single point of access for various asset types. The Liberty Repository provides early access to supported new content, including new product capabilities, when they are delivered, rather than waiting for a new release.

The Liberty Repository also enables fast and simple integration with open source, more refined access to runtime capabilities, and quick access to configuration and administration resources for developers and operations teams.

Important: Product documentation that is marked with the  icon indicates information about assets that are available only from the Liberty Repository.

Assets

Asset types available from the Liberty Repository are as follows:

Addons

Artifacts that are packaged to add new capabilities over an existing Liberty installation.

Admin Scripts

Sample scripts for common Liberty administrative tasks.

Config Snippets

Samples of Liberty server configurations for specific tasks.

Features

Individual units of server functionality that can be installed into the Liberty runtime environment.

Open Source Integration

Artifacts that provide simple Liberty integration with commonly used open source projects.

Products

Simple archive installation packages of the Liberty server runtime environment.

Product Samples

Sample server applications that demonstrate the use of Liberty runtime capabilities.

Tools Tools to enable development and test of Liberty-based applications and runtime extensions.

Features

The Liberty Repository enables immediate access to fully supported and production-ready features, without the need to wait for new product releases. The features that you add inherit the same support as your existing installation. For a full list of features that are available from the Liberty Repository, see “Liberty features” on page 483.

8.5.5.6

Offline access to Liberty Repository assets

In addition to accessing assets in the public, online Liberty Repository, you can create the following types of repositories to enable on-premises or offline access to Liberty Repository assets:

Liberty Asset Repository Service

An open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. To get started with the service, see the WASdev/tool.lars project on GitHub.

Local directory-based repository

Local directory-based repositories that you create when you download assets by using the

installUtility download command. For more information about the command, see “Downloading assets using the installUtility command” on page 860.

You can populate these repositories with your own customized content or download Liberty Repository assets by using the **installUtility download** command. As an alternative to downloading individual assets, you can download and extract a `wlp-featureRepo-<version>.zip` file from IBM Fix Central. The .zip file contains a directory-based repository of all features and add-ons for the particular fix pack of the Liberty. After you populate the repositories, you can install assets from them by using IBM Installation Manager or the **installUtility install** command.

8.5.5.8 For Version 8.5.5.8 and later, you can use Installation Manager and the **installUtility** command to work with repository assets in an archive file. You can access assets directly from the `wlp-featureRepo-<version>.zip` file and your own compressed directory-based repositories without extracting the archives.

Ways to access the Liberty Repository

You can access the Liberty Repository in the following ways:

WASdev.net website

You can access the Liberty Repository from the Downloads page on WASdev.net, an IBM developerWorks® website for Java application developers. You can browse and download content, filter by asset type, filter by product edition, and search on asset title, description, or edition.

Click any asset from the results of your search, filters, or both to take you to the asset details page. The asset details page provides an asset summary, a description of the asset, installation instructions, and configuration instructions. Links to related assets are also included with compatibility information.

Developer tools

The Liberty Repository is integrated in developer tools. These tools allow:

- Installation of the server runtime environment from an archive
- Installation of additional content, including features, samples and open source integrations
- Post-installation content
- Runtime and feature dependency awareness, searching and filtering, and configuration snippet inclusion

Installers

Assets can be installed from the Liberty Repository by using IBM Installation Manager and command-line utilities.

- IBM Installation Manager: If you have IBM Installation Manager installed, you can download and install assets from the Liberty Repository when you install IBM WebSphere Application Server V8.5.5.2 Liberty or later, or upgrade from a previous version of Liberty. In Version 8.5.5.6 or later, you can also install assets from a local directory-based repository or an instance of the Liberty Asset Repository Service.
- Command-line utilities:
 - **8.5.5.6** **installUtility**: Find, obtain information about, and install assets that are in a local directory-based repository, an instance of the Liberty Asset Repository Service, or the public Liberty Repository.
 - **featureManager**: Install a feature from the public Liberty Repository and obtain details of features that are already installed.

For more information on the different methods available for installing Liberty Repository assets, see “Installing Liberty Repository assets” on page 852.

To access the IBM WebSphere Liberty Repository with limited internet access or through a firewall, ensure that you have access to the following hosts and ports:

- public.dhe.ibm.com on port 443
- asset-websphere.ibm.com on port 443

Shared libraries

Shared libraries are files used by multiple applications. You can use shared libraries and global libraries to reduce the number of duplicate library files on your system.

Library elements

Liberty libraries have three elements; <folder>, <file>, and <fileset>. For example:

```
<library>
  <folder dir="..." />
  <file name="..." />
  <fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>
```

A specified file must be a container for the resource (for example a JAR file) rather than the resource itself.

If an element in the list is a file, the contents of that JAR or compressed .zip file are searched. If a folder is specified then resources are loaded from that directory.

Global libraries

Global libraries can be used by any application. JAR files are placed in a global library directory, and then are specified in the class loader configuration for each application.

You can place global libraries in two locations:

- \${shared.config.dir}/lib/global
- \${server.config.dir}/lib/global

If there are files present in these locations at the time an application is started, and that application does not have a <classloader> element configured, the application uses these libraries. If a class loader configuration is present, these libraries are not used unless the global library is explicitly referenced.

For more information, see “Providing global libraries for all Java EE applications” on page 978.

Resource files

Within Liberty libraries, you can have resource files defined in the library element. For example,

```
<library>
  <folder dir="..." />
  <file name="..." />
  <fileset dir="..." includes="*.jar" scanInterval="5s" />
  <folder dir="${server.config.dir}/mylibs" />
  <file name="${server.config.dir}/otherlibs/my.jar" />
</library>
```

The folder setting in the previous example, allows all files under the mylibs directory to be available on the classpath. You can use this style of entry to have your .xml and .properties available.

Library elements

Liberty libraries have three child elements, <folder>, <file> and <fileset>. For example,


```

<library>
  <folder dir="..." />
  <file name="..." />
  <fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>

```

- <folder>: All resources under each configured folder will be loadable
- <file>: Each configured file should be either a native library or a container for resources (such as a JAR or a ZIP file). All resources within a container are loadable and any other filetype that is specified will have no effect.
- <fileset>: Each configured fileset is effectively a collection of files. Each file in the fileset should be a native library or a container for resources (such as a JAR or a ZIP file). All resources within a container are loadable and any other filetype that is specified will have no effect.

For example,

```

<library id="someLibrary">
  <!-- Location of XML and .properties files in the file system for easy editing -->
  <folder dir="${server.config.dir}/editableConfig" />

  <!-- Location of some classes and resources in the file system -->
  <folder dir="${server.config.dir}/extraStuff" />

  <!-- A zip file containing some resources -->
  <file name="${server.config.dir}/lib/someResources.zip" />

  <!-- All the jar files in their servers lib folder -->
  <fileset dir="${server.config.dir}/lib" includes="*.jar" scanInterval="5s" />
</library>

<application location="webStore.war">
  <classloader commonLibraryRef="someLibrary" />
</application>

```

The configuration snippet in the previous example, allows all resources under the `editableConfig` directory to be loaded by the `webStore` application.

Product extension

You can expand the capability of Liberty by writing product extensions.

You can write your own Liberty features and install them onto an existing Liberty server, or you can package them for delivery to your users.

Liberty provides various System Programming Interfaces (SPIs) that you can use to extend the runtime environment; you can also use more advanced features such as operating the Liberty server from your Java applications programmatically.

Product extension

A product extension is a directory on disk that is structured like the Liberty installation directory, `${wlp.install.dir}`. It is defined to the Liberty installation through a file in the `${wlp.install.dir}/etc/extensions` directory called `extension-name.properties`. The contents of the product extension directory are then used to extend Liberty. Multiple product extensions can be installed together but they must have unique names; this is enforced through the naming of the properties files. The default product extension location, `${wlp.user.dir}/extension`, does not require a properties file; content under this location is automatically detected and is known to the runtime as the “usr” product extension.

Product extensions usually contain one or more Liberty features but can have any content that extends the Liberty environment, for example scripts or resources.

Best practice: Install your product extensions into directories that are not affected by updates to the Liberty environment. For more information, see “What might be modified by applying service or an upgrade?” on page 15.

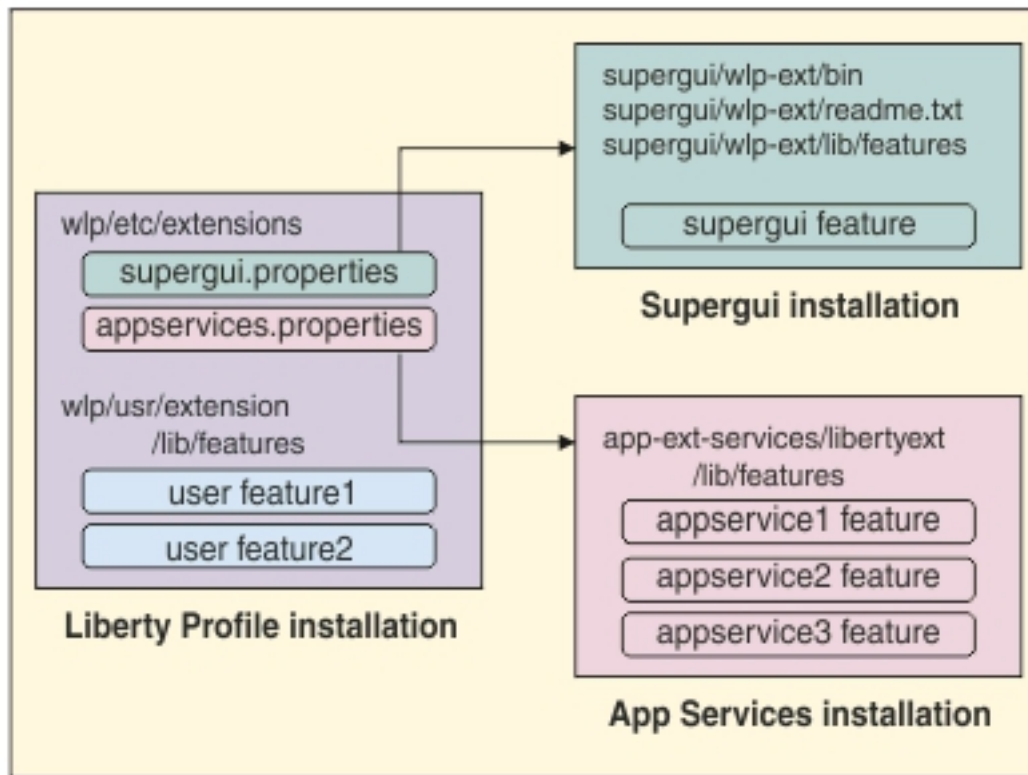


Figure 5.

The product extension file has the following properties:

```
com.ibm.websphere.productId=your_product_id
com.ibm.websphere.productInstall=absolute_or_relative_file_path
```

Note: When a relative file path is used, it must be a peer of the `${wlp.install.dir}` value.

For example:

```
com.ibm.websphere.productId=org.acme.supergui
com.ibm.websphere.productInstall=supergui/wlp-ext
```

Feature

A Liberty feature consists of a definition file (feature manifest), and a collection of OSGi bundles that provide classes and services corresponding to a particular capability in the Liberty runtime environment.

Scenarios for using a Liberty feature instead of a regular application

Implementing a function as a Liberty feature instead of as an application might be appropriate in a number of scenarios. The following list describes some of the benefits of using a feature:

- Features are controlled through feature manager configuration, so they are separate from user application management.
- Feature code has access to the Liberty SPI, which allows deeper integration with the runtime environment.

- Features can receive user-specified configuration from the `server.xml` file, and expose their configuration settings in the development tools without the tools having to be changed.
- Features can easily expose classes and services to each other and to user applications.
- Features can be extremely lightweight with no application container dependencies.
- Features can be used to augment a particular programming model. For example, a User Feature can add support for custom Blueprint namespace handlers or Blueprint annotations to the OSGi Application programming model.

Note: Features cannot generally be directly portable to other application servers; if portability is important you should use specification-compliant applications.

Developing a simple feature

See “Developing a Liberty feature manually” on page 1095, “Creating a Liberty feature by using developer tools” on page 1106, and “Liberty feature manifest files” on page 1097.

Using a feature in the server

To use a user-written feature in the Liberty server, you must install it into a product extension directory. This might be the predefined “user product extension” location or an extension that is located outside the Liberty installation directory. Then you can add the feature name into the server configuration.

The user product extension is a predefined directory where the server looks for additional features. The feature definition `.mf` file must be copied into the `${wlp.user.dir}/extension/lib/features` directory and the bundle `.jar` files must be copied into the `${wlp.user.dir}/extension/lib` directory.

User-written features are added to the server configuration in the same way as product features. To avoid name clashes with features from different providers, features that are part of a product extension must be prefixed with the extension name. For features in the `usr/extension/lib` directory, the default prefix is `usr:`.

For example, if you have installed a feature called `simple-1.0` into the `${wlp.user.dir}/extension/lib` directory, you must configure that feature into the `server.xml` as follows:

```
<featureManager>
  <feature>usr:simple-1.0</feature>
</featureManager>
```

If you have installed a feature called `myFeature` into your own location, and defined a product extension in the `${wlp.install.dir}/etc/extensions/myExt.properties` file, you must configure that feature into the `server.xml` file as follows:

```
<featureManager>
  <feature>myExt:myFeature</feature>
</featureManager>
```

When you start the server, the feature is detected by the feature manager and the bundles are installed into the OSGi framework and started.

See also “Adding and removing Liberty features” on page 968 and “Feature management” on page 482.

Programmatically using a feature from applications

Features can expose classes and services to applications.

To expose Java classes for application use, you must list the class packages in the `IBM-API-Package` header in the feature manifest. Listing the class packages in the `IBM-API-Package` header makes the classes visible

to the application class loaders. Visibility of API packages can be controlled through the API visibility type. See “Specifying API and SPI packages for a Liberty feature project” on page 1107.

To allow services to be used by OSGi applications, you must list them in the IBM-API-Service header in the feature manifest. A feature provides OSGi services so that you can refer to those services programmatically from your applications.

Services should generally be registered into the OSGi Service Registry (SR) to allow applications (or other features) to locate them. OSGi applications and other features can perform a direct lookup from the SR, or can use capabilities such as Blueprint or OSGi Declarative Services to inject their service dependencies. Java EE applications are more likely to locate services through JNDI; in Liberty there is a federation of the SR and JNDI that allows Java EE applications to use JNDI to locate services in the SR. For more information, see “Working with the OSGi service registry” on page 1112.

Exposing a feature as a web application

To expose a Liberty feature as a web application, you can publish the OSGi bundles in the feature as web application bundles (WABs). In addition to the OSGi headers that a bundle requires, you can specify the web application context path by using the Web-ContextPath header.

For example:

```
Web-ContextPath: myWABapp
Bundle-ClassPath: WEB-INF/classes
```

Configuration injection and processing

A major benefit of using features is that they can be easily configured by the user in the server.xml file (and included files). The configuration files are monitored and parsed by the Liberty profile kernel and the resulting sets of properties can be injected into the relevant component each time they change.

Liberty configuration is managed by the OSGi Configuration Admin (CA) service in the kernel and can be accessed according to that specification. Sets of configuration properties are identified by a persisted identity (PID) that is used to associate the element in the server.xml file with the component that registers to receive the properties.

For example, if you register your feature with the CA service using a PID of com.acme.console, a user can specify the following configuration in the server.xml file:

```
<com.acme.console color="blue" size="17"/>
```

And your feature will receive the properties:

- color="blue"
- size="17"

You can optionally provide metadata that describes your configuration properties by using OSGi Metatype descriptors. The use of descriptors causes your configuration metadata to be included in the configuration schema that is generated by Liberty and is used by the Developer Tools, so your configuration properties are automatically presented to application developers as they configure their server.

For more details on receiving and describing configuration properties, see “Enabling a service to receive configuration data” on page 1118.

Declarative Services in the Liberty

Larger or more complex features often benefit from the use of OSGi Declarative Services (DS) to enable the feature to be composed of multiple services, and to manage the injection of dependencies and configuration properties. The use of DS allows lazy activation of services, deferring the loading of Java classes until the service is used, and ordering the activation of services based on dependencies. Most of the features in the Liberty product are managed by DS.

See also “Composing advanced features by using OSGi Declarative Services” on page 1116.

Security

Liberty security provides protection for web resources in accordance with the Servlet 3.0 specification and EJB resources in accordance with the ejbLite 3.1 specification. The Liberty security also provides protection for the JMX connections when you are using the REST connector.

The following diagram shows a typical security process involved when accessing a protected web resource. To make the security process work, you must configure the appropriate security features and the configurations that are required for the authentication and authorization.

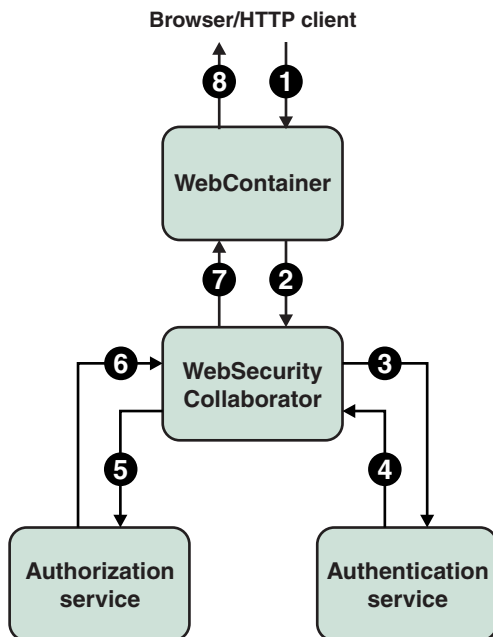


Figure 6. Typical security flow for web resources

1. An HTTP client requests a web resource in the WebContainer.
2. The WebContainer delegates the security check to the WebSecurity Collaborator.
3. The WebSecurity Collaborator prompts the user to enter credentials if absent, and uses the Authentication service to authenticate the user.
4. The Authentication service authenticates, creates, and returns the subject if authenticated successfully. Otherwise, the Authentication service reports an exception for the authentication failure.
5. The WebSecurity Collaborator uses the Authorization service to perform a user authorization check.
6. The Authorization service returns the authorization result to the WebSecurity Collaborator.
7. The WebSecurity Collaborator returns the result of the security check about whether the user is authorized.
8. The WebContainer serves or rejects the requested resource.

The following sections provides a summary of the primary security components in Liberty:

- “Quick start”
- “Authentication”
- “Authorization”
- “Secure Socket Layer (SSL)”
- “Single Sign-On (SSO)”
- “Web security-related properties” on page 583
- “Security public APIs” on page 583
- “Management security” on page 583
- “Authentication aliases” on page 583
- “Configuration examples and samples” on page 583
- “Security compatibility and differences” on page 583
- “Configuring Lightweight Directory Access Protocol (LDAP)” on page 583
- “Troubleshooting” on page 584
- **Distributed operating systems** “Tools” on page 584

Quick start

With the `quickStartSecurity` element, you can configure a single user security environment in Liberty. See “Quick overview of security” on page 584 for details of how the security workflow is when you use the `quickStartSecurity` element, and “Getting started with security in Liberty” on page 1147 for a sample task.

Authentication

Authentication confirms the identity of a user. The most common form of authentication is user name and password, such as through either basic authentication or form login for web applications. When a user is authenticated, the source of a request is represented as a Subject object at the run time. This process involves performing access control checks when a user accesses a resource, based on the authorization rules configured for the resource. See “Authentication” on page 585 for more concepts and “Authenticating users in Liberty” on page 1168 for detailed tasks.

Authorization

Authorization determines whether to grant a user access to resources within the system. The Java EE model uses subjects, resources, and roles to determine what can and cannot be allowed. This process involves checking the user credentials such as the user ID and password, certificates, and tokens, and creating a subject based on the authenticated user. See “Authorization” on page 606 for more concepts and “Authorizing access to resources in Liberty” on page 1254 for detailed tasks.

Secure Socket Layer (SSL)

SSL provides transport level security. See “Enabling SSL communication in Liberty” on page 1152 for detailed tasks.

Single Sign-On (SSO)

SSO enables access to applications without the user being prompted to login multiple times. See Concept of SSO for more details and “Customizing SSO configuration using LTPA cookies in Liberty” on page 1203 for the detailed task.

Web security-related properties

There are many configuration properties that you can configure as part of web security, such as SSO and client certificate authentication, for your applications. See ****** MISSING FILE ****** for available attributes and see “Configuring web security related properties in Liberty” on page 1297 for some examples.

Security public APIs

Liberty contains public APIs that you can use to implement security functions. The security public APIs in Liberty are a subset of the traditional security public APIs. The main classes are `WSSecurityHelper`, `WSSubject`, and `RegistryHelper`. These classes contain a subset of the methods that are available in the WebSphere Application Server traditional versions. There is also a new class `WebSecurityHelper`. See “Security public APIs” on page 611.

The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate `.zip` file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

See “Developing extensions to the Liberty security infrastructure” on page 1301 for some examples.

Management security

Management security means that you can manage Liberty by using a remote JMX client. To secure remote connections using the REST connector, see “Connecting to Liberty by using JMX” on page 1021. You can also develop your own JMX client application as described in “Developing a JMX Java client for Liberty” on page 1024.

Authentication aliases

Authentication data aliases provide the security support for database connectivity. See “Configuring authentication aliases for Liberty” on page 1300.

Configuration examples and samples

There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty.

Security compatibility and differences

You can learn about the main differences in the security capability between the traditional and Liberty. See “Configuration differences between the traditional and Liberty: security” on page 615.

Configuring Lightweight Directory Access Protocol (LDAP)

After selecting the LDAP User Registry item to add to the server configuration, the **LDAP User Registry Details** panel will display a list for the supported LDAP server types. If you select a supported LDAP server type, the LDAP filters associated with the selected LDAP server type will not automatically pre-populated.

Each of the supported LDAP server types has a default set of filters defined. After the **LDAP User Registry** item and server type has been added, the associated LDAP filters can be configured by selecting the **LDAP User Registry** configuration and adding the required LDAP filter:

- Active Directory LDAP filters
- Custom LDAP filters
- Domino LDAP filters

- eDirectory LDAP filters
- IBM Directory Server LDAP filters
- iPlanet LDAP filters
- Netscape LDAP filters
- SecureWay LDAP filters

Selecting any of the LDAP filters will display the default values for the filter types:

- user filter
- group filter
- user ID map
- group ID map
- group member ID map

If the default filters are used, the `server.xml` file is not updated with any filter information. If any of the filters are changed, only the changed filter types will be updated in the `server.xml`.

Note: If you do not specify or select a reference ID using the **Browse** button, the default filters associated with selected LDAP server type will be used.

Alternatively, you can add an LDAP filter to the server configuration. An ID must be specified to associate the reference to this particular filter configuration, in order to associate it with the LDAP User Registry configuration. If this method of configuring the LDAP filters is used, the reference ID will then be selected on the **LDAP User Registry Details** panel (located using the **Browse** button under the respective LDAP filter type).

If you are using Eclipse-based developer tools to configure LDAP, verify the configuration saved against the samples in `wlp/templates/config/ldapRegistry.xml`.

For more information, see “Configuring LDAP user registries in Liberty” on page 1169.

Troubleshooting

Use the troubleshooting information to solve security-related problems when you use Liberty. See “Troubleshooting security” on page 1452 and “Troubleshooting LDAP” on page 1454.

Distributed operating systems

Tools

Configure security by using the Eclipse-based developer tools for Liberty. See “Editing the Liberty configuration by using developer tools” on page 938. Specific information about tools and security configuration is available in “Configuring TAI on Liberty by using developer tools” on page 1206 and “Configuring JAAS on Liberty by using developer tools” on page 1186.

Quick overview of security

To understand the basic workflow of security in Liberty, some common security terms are detailed along with an example.

Security key terms

Authentication

Authentication confirms the identity of a user. The most common form of authentication is user name and password, such as through basic authentication or form login for web applications. When a user is authenticated, the source of a request is represented as a Subject object at run time.

Authorization

Authorization determines whether a user has access to a given role within the system. The Java EE model uses subjects, roles, and role mappings to determine if access is allowed.

Role A role is defined within the Java EE application. Some roles, such as the Administrator role, are predefined by the system. Other roles are defined by the application developer. In Java EE, subjects are usually granted or denied access to a role based on the roles they perform within the application.

Subject

A subject is both a general term and a Java object: `javax.security.auth.Subject`. Generally, the term subject means active entities within the system, such as users on the system, and even the system process itself.

Security workflow example

The following example demonstrates how the security is applied when a user requests access to a resource. For example, a user Bob wants to access a servlet `myWebApp`. See the code samples in “Getting started with security in Liberty” on page 1147.

To access the servlet `myWebApp`, the following conditions must be true:

1. Bob must be able to log in to the system because the servlet is protected.
2. Bob must be in the testing role because the servlet is restricted by using an `auth-constraint` element in the deployment descriptor.

If Bob cannot log in to the system, or Bob is not in the testing role, then the access to the servlet `myWebApp` is denied.

Another user Alice can log in to the system because Alice is a valid user. But Alice is not in the testing role. An HTTP 403 error (Access Denied/Forbidden) displays when Alice logs in.

Authentication

Authentication in the Liberty security is to confirm the identity of a user.

To access a protected web resource, the user must provide credential data, such as user ID and password. The authentication process involves collecting this user credential information (based on how the web application was configured to collect this data) and validating it against the configured registry. When the credential information is verified, a JAAS subject is created for that user. The subject contains additional information about the user, such as the groups that the user belongs to, and the tokens created for the user. The information in this subject is then used during the authorization process to determine whether the user can access the resource.

The following diagram illustrates a typical authentication process flow for a web resource.

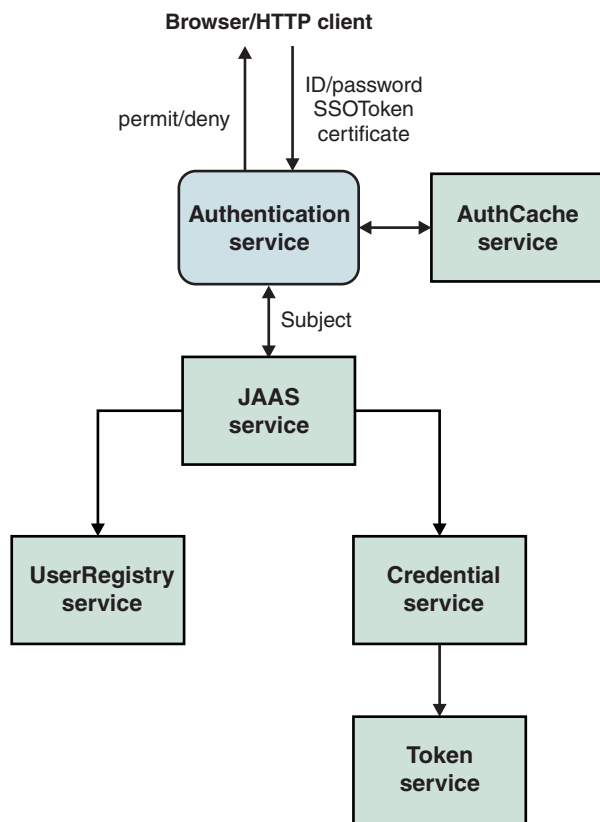


Figure 7. Overview of authentication process




The authentication process involves gathering credential data from the user, checking the cache to see whether the subject exists for that user and in its absence calling the JAAS service to perform the authentication to create a subject. The JAAS service calls a set of login modules to handle the authentication. One or more of the login modules creates the subject depending on the credential data. The login module then calls the registry that is configured to validate the credential information. If the validation is successful, the authentication process collects and creates relevant information for that user, including the groups that the user belongs to and the single sign-on (SSO) token used for SSO capability, and stores them in the subject as relevant credentials. You can also customize the information saved in the subject by plugging in custom login modules during this process.

When the authentication is successful, the SSO token that is created during the process is sent back to the browser in a cookie. The default name of the configurable cookie is `1tpaToken2`. On subsequent calls, the token information is used to authenticate the user. If this authentication fails, the authentication service tries to use other authentication data, such as the user ID and password, if they still exist in the request.

Note: To support user IDs and passwords that contain non US-ASCII characters, form login method is required for web applications. For more information, see `autoRequestEncoding` and `autoResponseEncoding`.

The following sections describe these concepts in detail:

- “User registries” on page 587
- “Authentication cache” on page 587
- “JAAS configuration” on page 587
- “JAAS login modules” on page 588

- “Callback handler” on page 589
- “Credentials and tokens” on page 589
- “LTPA” on page 589
-  **8.5.5.5** “SPNEGO” on page 590
-  **8.5.5.8** “Kerberos constrained delegation for SPNEGO” on page 591
- “Single sign-on (SSO)” on page 591
-  **8.5.5.7** “SAML Web Browser SSO” on page 591
- “Pluggable authentication” on page 592
- “Identity assertion” on page 592
- “RunAs() authentication” on page 593
- “Proxy login module” on page 593
- “Certificate login” on page 594
- “Hash table login module” on page 594

User registries

When validating the authentication data of a user, the login modules call the user registry that is configured to validate the user information. Liberty supports both a simple configuration-based user registry and a more robust LDAP-based registry. For more information, see “Configuring a user registry for Liberty” on page 1168.

Using the LDAP registry, you can also federate multiple repositories and execute the LDAP operations on two or more registries. The Liberty user can configure the LDAP registry federation feature either directly in the `server.xml` file or can configure in the **LDAP Registry Federation** section in the developer tool. After the configuration of the federated repositories, you can obtain a consolidated result of the federated repositories on any operation that you want to perform. For example, if you want to perform a search operation for all user names that starts with *test*, you can perform a search across the set of LDAP registries and get the consolidated search result which can then be sent back to the calling program.

Authentication cache

Because creating a subject is relatively expensive, Liberty provides an authentication cache to store a subject after the authentication of a user is successful. The default expiration time for the cache is 10 minutes. If the user does not log back in within 10 minutes, the subject is removed and the process of authentication repeats to create a subject for that user. Changes to the configuration that affect the creation of the subject, such as adding a login module or changing the LTPA keys, will cause the authentication cache to be cleared. If the subject is cached and the information in the registry changes, the cache is updated with the information in the registry. You can configure the cache timeout period, and the cache size, and you can also disable or enable caching. For more information, see “Configuring the authentication cache in Liberty” on page 1183.

JAAS configuration

JAAS configuration defines a set of login modules to create the subject. Liberty supports the following JAAS configurations:

system.WEB_INBOUND

Used when accessing web resources such as servlets and JSPs.

WSLogin

Used by applications when using the programmatic login. It is also used by applications running

in an application client container, but unlike the `ClientContainerJAAS` configuration, it does not recognize the `CallbackHandler` handler that is specified in the client application module's deployment descriptor.

system.DEFAULT

Used for login when no JAAS configuration is specified.

8.5.5.4 system.DESERIALIZE_CONTEXT

Used when a security context is being deserialized. This JAAS configuration handles authentication to re-create the subjects that were active on the thread at the time the security context was serialized. You can specify this JAAS configuration and add your own custom JAAS login modules by editing the JAAS configuration entry in the `server.xml` file to ensure that the propagated subjects contain your custom information.

8.5.5.6 ClientContainer

Used by applications running in an application client container. This JAAS login configuration recognizes the `CallbackHandler` handler that is specified in the client application module's deployment descriptor, if one is specified.

The `system.WEB_INBOUND` and `system.DEFAULT` configurations have these default login modules in this order: **hashtable**, **userNameAndPassword**, **certificate**, and **token**. The `WSLogin` configuration has the proxy login module as the default login module, and the proxy delegates all the operations to the real login module in `system.DEFAULT`.

No explicit configuration is required unless you want to customize by using the custom login modules. Depending on the requirement, you can customize specific login configurations. For example, if you want all the web resource logins to be customized, you must add custom login modules only to the `system.WEB_INBOUND` configuration. See “Configuring a JAAS custom login module for Liberty” on page 1184.

JAAS login modules

JAAS configuration uses a set of login modules to create the subject. Liberty provides a set of login modules in each of the login configurations. Depending on the authentication data, a particular login module creates the subject. The authentication data is passed to the login modules by using the callback handler, as specified in the JAAS specification. For example, if the user ID and password callback handler is being used for authentication, the **userNameAndPassword** login module handles the authentication. If a `SingleSignOnToken` credential is presented as the authentication data, only the **token** login module handles the authentication.

The following default login modules are supported in Liberty:

userNameAndPassword

Handles the authentication when user name and password are used as the authentication data.

certificate

Handles the authentication when an X509 certificate is used as the authentication data of mutual SSL.

token Handles the authentication when an SSO token is presented as the authentication data. During the authentication process, an SSO token is created and sent back to the HTTP client (browser) in a cookie. On subsequent requests, this cookie is sent back by the browser and the server extracts the token from the cookie to authenticate the user when the single sign-on is enabled.

hashtable

Used when the authenticated data is sent through a predefined hash table. For more information about the hash table login, see “Hash table login module” on page 594. This login module is also used by the security run time when authentication is performed using identity only; for example, in the case of `runAs`.

proxy The default login module for WSLogin. See “Proxy login module” on page 593.

The login modules are called in the order that they are configured. The default order is **hashtable**, **userNameAndPassword**, **certificate**, **token**. If you must customize the login process by using custom login modules, you can provide them and configure them in the order you need. Typically, place a custom login module first in the list of login modules so that it is called first. When a custom login module is used, you must specify all the login module information in the configuration along with the custom login module in the required order.

When a login module determines that it can handle the authentication, it first makes sure that the authentication data that is passed in is valid. For example, for user name and password authentication, the configured user registry is called to verify the authentication information. For token authentication, the token must be decrypted and valid for the verification to succeed.

When the authentication data is validated, the login modules create credentials with additional data for the user including the groups and the SSO token. A custom login module can add additional data to the subject by creating its own credentials. For the Liberty authorization to work, the subject must contain the `WSCredential`, `WSPrincipal`, and `SingleSignonToken` credentials. The `WSCredential` credential contains the groups information, with additional information that is required by the security runtime environment.

Callback handler

The Liberty supports various callback handlers for providing data to the login modules during the JAAS authentication process. A custom login module can use the callback handler information to authenticate itself. For example, if the callback handler needs to access some information in an `HttpServletRequest` object, it can do so by using that specific callback handler.

The following callback handlers and factories for programmatic JAAS login are supported in Liberty:

- `com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl`
- `com.ibm.wsspi.security.auth.callback.WSCallbackHandlerFactory`

The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

See “Developing JAAS custom login modules for a system login configuration” on page 1305.

Credentials and tokens

As mentioned in the `loginModule` section, credentials are created as part of the subject creation process. Liberty creates the `WSCredential`, `SingleSignonToken`, and `WSPrincipal` credentials. The `SingleSignonToken` credential contains the token that is sent back to the browser in a cookie for SSO to work. This token contains the user information and an expiration time. It is signed and encrypted by using the Lightweight Third Party Authentication (LTPA) keys that are generated during the first server startup. The default expiration time is 2 hours and is an absolute time, not based on user activities. After the 2 hours, the token expires and the user must log in again to access the resource.

LTPA

LTPA is intended for distributed and multiple application server environments. In Liberty, LTPA supports SSO and security in a distributed environment through cryptography. This support enables LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature.

Application servers can securely communicate using the LTPA protocol. The protocol also provides the SSO feature, whereby a user is required to authenticate only when connecting to a domain name system

(DNS). Then the user can access resources in other Liberty servers in the same domain without getting prompted. The realm names on each system in the DNS domain are case-sensitive and must match identically.

The LTPA protocol uses cryptographic keys to encrypt and decrypt user data that passes between the servers. These keys must be shared between different servers for the resources in one server to access resources in other servers, assuming that all the servers involved use the same user registry. LTPA requires that the configured user registry must be a centrally shared repository so that users and groups are the same, regardless of the server.

When using LTPA, a token is created that contains the user information and an expiration time, and is signed by the keys. The LTPA token is time sensitive. All participating servers must have their time and date synchronized. If not, LTPA tokens are prematurely expired and cause authentication or validation failures. Coordinated Universal Time (UTC) is used by default, and all other servers must have the same UTC time. See your operating system documentation for information about how to ensure the same UTC time among servers.

The LTPA token passes to other servers through cookies for web resources when SSO is enabled.

If the receiving servers use the same keys as the originating server, the token can be decrypted to obtain the user information, which then is validated to make sure that the token is not expired and that the user information in the token is valid in its registry. On successful validation, the resources in the receiving servers are accessible after the authorization check.

Each server must have valid credentials. When the credentials expire, the server is required to communicate to the user registry to authenticate. Extending the time that the LTPA token remains cached presents a slightly increased security risk to be considered when defining your security policies.

If key sharing is required between different Liberty servers, copy the keys from one server to another. For security purposes, the keys are encrypted with a randomly-generated key, and a user-defined password is used to protect the keys. This same password is needed when importing the keys into another server. The password is used only to protect the keys, and is not used to generate the keys.

When security is enabled, LTPA is configured by default during the Liberty server start time. For more information about the LTPA support, see “Configuring LTPA in Liberty” on page 1188.



8.5.5.5

SPNEGO

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) enables users to log in to the Microsoft Domain controller once and access protected applications on Liberty servers without getting prompted again.

When SPNEGO web authentication is enabled, and the browser client accesses a protected resource on the Liberty server, SPNEGO is responsible for authenticating access to the secured resource that is identified in the HTTP request. The browser client creates a SPNEGO token and sends it to the Liberty server as part of the HTTP request. The WebSphere Application Server validates and retrieves the user identity from the SPNEGO token. The identity is used to establish a secure context between the user and the application server.

For more information about SPNEGO, see  8.5.5.5 SPNEGO. For further information on configuring SPNEGO on the Liberty server, see  8.5.5.5 Configuring SPNEGO authentication in Liberty.



8.5.5.8

Kerberos constrained delegation for SPNEGO

The Kerberos constrained delegation feature provides two APIs that are used to create out-bound SPNEGO tokens for back end services that support SPNEGO authentication, such as .NET servers and other Liberty servers.

The Kerberos v5 extension called S4U (Services for Users) comprises two parts:

S4U2self

Allows a Liberty server to obtain a service ticket to itself on behalf of a user. This can be used with any form of authentication that is supported by Liberty. S4U2self is the Kerberos Protocol Transition extension.

S4U2proxy

Allows a Liberty server to obtain service tickets to trusted services on behalf of a user. These service tickets are obtained by using the user's service ticket to the Liberty service. The services are constrained by the Kerberos Key Distribution Center (KDC) administrator. S4U2proxy is the Kerberos Constrained Delegation extension.

Single sign-on (SSO)

SSO enables users to log in in one place (one server for example) and access applications on other servers without getting prompted again. To make SSO work, the LTPA keys must be exchanged across different Liberty servers, the user registries must be the same, and the token must not have expired. To exchange the LTPA keys, you can copy the `ltpa.keys` file from one server to another and restart the server to use the new LTPA keys. The registries that are used by all the servers participating in the SSO domain must be the same.

When a user is authenticated in one Liberty server, the SSO token created for the user during the authentication process is put in the cookie that is sent to the HTTP client, for example a browser. If there is another request from that client to access another set of applications on a different server, but in the same DNS that was configured as part of the SSO configuration in the first server, the cookie is sent along with the request. The receiving server tries to authenticate the user using the token in the cookie. If both conditions are met, the receiving server validates the token and creates a subject based on the user in this server, without prompting the user to log in again. If the token cannot be validated (for example, it cannot decrypt or verify the token because of LTPA key mismatch), the user is prompted to enter the credential information again.

Any application that is configured to use the `Form-login` attribute must have SSO to be configured for that server. When the user is authenticated for a `form-login`, the token is sent back to the browser that will be used for authorizing the user when the resource is accessed.

See “Customizing SSO configuration using LTPA cookies in Liberty” on page 1203.



8.5.5.7

SAML Web Browser SSO

SAML Web Browser SSO enables web applications to delegate user authentication to a SAML identity provider instead of a configured user registry.

For further information on configuring SAML Web Browser SSO on the Liberty server, see “SAML 2.0 Web Browser Single-Sign-On” on page 603.

Pluggable authentication

Use the following ways to customize the authentication process:

- Provide a custom login module. Most of the authentication process is built around JAAS login modules, so you can plug in custom login modules before, after, or between the login modules provided by Liberty. See “Configuring a JAAS custom login module for Liberty” on page 1184.
- Implement Trust Association Interceptor (TAI) to handle all web resource-based authentication. See “Developing a custom TAI for Liberty” on page 1301.

For more details about the JAAS login module and TAI, see Advanced authentication in WebSphere Application Server.

Identity assertion

Besides authentication that requires a requesting entity to prove its identity, Liberty also supports identity assertion. This is a relaxed form of authentication that does not require identity proof, but rather accepts the identity based on a trust relationship with the entity that vouches for the asserted identity.

Use the following ways to assert identities in Liberty

1. Use the hash table login. See “Developing JAAS custom login modules for a system login configuration” on page 1305.
2. Use `IdentityAssertionLoginModule`. You can allow an application or system provider to assert an identity with trust validation. To use `IdentityAssertionLoginModule`, use the JAAS login framework, where trust validation is accomplished in one custom login module and credential creation is accomplished in `IdentityAssertionLoginModule`. You can use the two login modules to create a JAAS login configuration that can be used to assert an identity.

The following two custom login modules are required:

User implemented login module (trust validation)

The user implemented login module performs whatever the user requires in trust verification. When trust is verified, the trust verification status and the login identity must be put into a map in the share state of the login module, so that the credential creation login module can use the information. Store this map in the following property:

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

This property consists of the following properties:

- `com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted`

This property is set to true if trusted and false if not trusted.

- `com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal`

This property contains the principal of the identity.

- `com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates`

This property contains the certificate of the identity.

Identity assertion login module (credential creation)

The following module creates the credential:

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule
```

This module relies on the trust state information in the shared state of the login context.

The identity assertion login module looks for the trust information in the shared state property:

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

This property contains the trust status and the identity to log in, and must include the following property:

- ```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted
```

This property is set to true when trusted and false when not trusted.

- ```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal
```

This property contains the principal of the identity to log in if a principal is used.

- ```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates
```

This property contains an array of a certificate chain that contains the identity to log in if a certificate is used.

A `WSLoginFailedException` message is returned if the state, trust, or identity information is missing. The login module then logs in with the identity, and the subject contains the new identity.

See “Customizing an application login to perform an identity assertion by using JAAS” on page 1316.

## RunAs() authentication

When you have successfully authenticated after calling a servlet, the servlet can make subsequent calls, for example to other servlets. These subsequent calls are normally made under the same security identity that you originally used to log in to the servlet. This identity is known as the caller identity. Alternatively, you can choose to delegate to a different identity by using the RunAs specification, so that any subsequent calls that the servlet makes run under this other identity. To summarize, you have two options to propagate the security identity:

- Propagate the caller identity, which is the default behavior.
- Delegate to the RunAs identity that you can specify by using the RunAs specification.

After the server authenticates the original user, the server then authenticates the RunAs user. If this authentication fails, the server falls back to propagate the caller identity.

To use the RunAs specification, you must update the deployment descriptor of your application to include the `run-as` element or `@RunAs` annotation. Set this element to the security role that you want to delegate to.

See “Configuring RunAs authentication in Liberty” on page 1203.

## Proxy login module

The proxy login module class loads the application server login module and delegates all the operations to the real login module implementation. The real login module implementation is specified as the `delegate` option in the option configuration. The proxy login module is required because the application class loaders do not have visibility of shared library class loaders of the application server product. With an application programmatic login that uses the `Login()` method of the `LoginContext` class with JAAS login context entry `WSLogin`, the proxy login module delegates all the work to the JAAS login context entry `system.DEFAULT`.

## Certificate login

With the certificate login feature, you can authenticate web requests such as servlets by using client side X509 certificates instead of supplying a user ID and password.

Certificate authentication works by associating a user in the user registry with the distinguished name in the client certificate of a web request. Trust is established by having the client certificate trusted by the server, for example the signer of the client certificate must be in the trust store of the server. This mechanism eliminates the need for users to supply a password to establish trust.

See “Securing communications in Liberty” on page 1151.

## Hash table login module

Look up the required attributes from the user registry, put the attributes in a hash table, and then add the hash table to the shared state. If the identity is switched in this login module, you must add the hash table to the shared state. If the identity is not switched, but the value of the `requiresLogin` code is true, you can create the hash table of attributes. You do not have to create a hash table in this situation, because Liberty handles the login for you. However, you might consider creating a hash table to gather attributes in special cases. For example, if you are using your own special user registry, then creating a `UserRegistry` implementation, using a hash table, and letting the server gather the user attributes for you, might be a simple solution.

The following rules define in more details about how a hash table login is completed. You must use a `java.util.Hashtable` object in either the Subject (public or private credential set) or the shared-state `HashMap`. The `com.ibm.wsspi.security.token.AttributeNameConstants` class defines the keys that contain the user information. If the `Hashtable` object is put into the shared state of the login context using a custom login module that is listed before the hashtable login module, the value of the `java.util.Hashtable` object is searched using the following key within the shared-state `HashMap`:

### Property

`com.ibm.wsspi.security.cred.propertiesObject`

### Reference to the property

`AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY`

### Explanation

This key searches for the `Hashtable` object that contains the required properties in the shared state of the login context.

### Expected result

A `java.util.Hashtable` object.

If a `java.util.Hashtable` object is found inside the Subject or within the shared state area, verify that the following properties are present in the hash table:

- `com.ibm.wsspi.security.cred.uniqueId`

### Reference to the property

`AttributeNameConstants.WSCREDENTIAL_UNIQUEID`

### Returns

`java.util.String`

### Explanation

The value of the property must be a unique representation of the user. For the Liberty default implementation, this property represents the information that is stored in the application authorization configuration. The information is in the application deployment descriptor after it is deployed and the user-to-role mapping is performed. See the expected format examples if the user to role mapping is performed by looking up a user registry implementation of Liberty.

## Expected format examples

Table 7. Format examples for `uniqueId`. This table gives expected format examples.

| User Registry | Format (UniqueUserId) value                  |
|---------------|----------------------------------------------|
| LDAP          | ldapRegistryRealm/cn=kevin,o=mycompany,c=use |
| Basic         | basicRegistryRealm/kelvin                    |

The `com.ibm.wsspi.security.cred.uniqueId` property is required.

- `com.ibm.wsspi.security.cred.securityName`

### Reference to the property

`AttributeNameConstants.WSCREDENTIAL_SECURITYNAME`

### Returns

`java.util.String`

### Explanation

This property searches for **securityName** of the authentication user. This name is commonly called the display name or short name. Liberty uses the **securityName** attribute for the `getRemoteUser`, `getUserPrincipal`, and `getCallerPrincipal` application programming interfaces (APIs). To ensure compatibility with the default implementation for the **securityName** value, call the public `String` `getUserSecurityName(String uniqueUserId)` `UserRegistry` method.

## Expected format examples

Table 8. Format examples for `securityName`. This table gives expected format examples.

| User Registry | Format (securityName) value |
|---------------|-----------------------------|
| LDAP          | kevin                       |
| Basic         | kevin                       |

The `com.ibm.wsspi.security.cred.securityname` property is required.

- `com.ibm.wsspi.security.cred.group`

### Reference to the property

`AttributeNameConstants.WSCREDENTIAL_GROUP`

### Returns

`java.util.ArrayList`

### Explanation

This key searches for the array list of groups to which the user belongs. The groups are specified in the `realm_name/user_name` format. The format of these groups is important because the groups are used by the Liberty authorization engine for group-to-role mappings in the deployment descriptor. The format that is provided must match the format expected by the Liberty default implementation. When you use a third-party authorization provider, you must use the format that is expected by the third-party provider. To ensure compatibility with the default implementation for the unique group IDs value, call the public `List` `getUniqueGroupIds(String uniqueUserId)` `UserRegistry` method.

## Expected format examples

Table 9. Format examples for `group`. This table gives some format examples when configuring inbound identity mapping.

| User Registry | Format (group) value                      |
|---------------|-------------------------------------------|
| LDAP          | ldapRegistryRealm/cn=group1,o=Groups,c=US |
| Basic         | basicRegistryRealm/group1                 |

The `com.ibm.wsspi.security.cred.group` property is required. A user is not required to be associated groups.

- `com.ibm.wsspi.security.cred.cacheKey`

#### Reference to the property

`AttributeNameConstants.WSCREDENTIAL_CACHE_KEY`

#### Returns

`java.lang.Object`

#### Explanation

This key property can specify an object that represents the unique properties of the login, including the user-specific information and the user dynamic attributes that might affect uniqueness. For example, when the user logs in from location A, which might affect their access control, the cache key must include location A so that the received Subject is the correct Subject for the current location.

This `com.ibm.wsspi.security.cred.cacheKey` property is not required. When this property is not specified, the cache lookup is the value that is specified for `WSCREDENTIAL_UNIQUEID`. When this information is found in the `java.util.Hashtable` object, Liberty creates a Subject similar to the Subject that goes through the normal login process, at least for LTPA. The new Subject contains a `WSCredential` object and a `WSPrincipal` object that is fully populated with the information found in the `Hashtable` object.

## Single sign-on for HTTP requests using SPNEGO web authentication



8.5.5.5

You can securely negotiate and authenticate HTTP requests for protected resources in the WebSphere Application Server by using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) as the web authentication service for WebSphere Application Server.

The following sections describe SPNEGO web authentication in more detail:

- “What is SPNEGO?”
- “SPNEGO web authentication in a single Kerberos realm” on page 597
- “SPNEGO web authentication in trusted Kerberos realms” on page 598
- “Support information for SPNEGO web authentication with a browser client” on page 600

### What is SPNEGO?

SPNEGO is a standard specification that is defined in The Simple and Protected GSS-API Negotiation Mechanism (IETF RFC 2478).

When Liberty server security is enabled, and SPNEGO web authentication is enabled, SPNEGO is initialized when processing a first inbound HTTP request. When the authentication filter is not specified, or the authentication filter is specified and the criteria is met, SPNEGO is responsible for authenticating access to the protected resource that is identified in the HTTP request.

In addition to WebSphere Application Server security runtime services, some external components are required to enable the operation of SPNEGO. These external components include:

- **Windows** Microsoft Windows Servers with Active Directory domain and associated Kerberos Key Distribution Center (KDC).
- A client application, for example, Microsoft .NET, or web service and J2EE client that supports the SPNEGO web authentication mechanism, as defined in IETF RFC 2478. Microsoft Internet Explorer and Mozilla Firefox are browser examples. Any browser that is being used must be configured to use the SPNEGO web authentication mechanism.

The authentication of HTTP requests is triggered by the user (the client-side), which generates an SPNEGO token. WebSphere Application Server receives this token. Specifically, the SPNEGO web authentication decodes and retrieves the user identity from the SPNEGO token. The identity is then used to make authorization decisions.

SPNEGO web authentication is a server-side solution in the WebSphere Application Server. Client-side applications are responsible for generating the SPNEGO token for use by SPNEGO web authentication. The user identity in the WebSphere Application Server security registry must be identical to the identity that the SPNEGO web authentication retrieves. An identical match does occur when Microsoft Windows Active Directory server is the Lightweight Directory Access Protocol (LDAP) server that is used in WebSphere Application Server. A custom login module is available as a plug-in to support custom mapping of the identity from the Active Directory to the WebSphere Application Server security registry.

WebSphere Application Server validates the identity against its security registry. If the validation is successful, the client GSS delegation credential is retrieved and placed in the client subject, and a Lightweight Third Party Authentication (LTPA) security token is created. It then returns the LTPA cookie to the user in the HTTP response. Subsequent HTTP requests from this same user to access more protected resources in the WebSphere Application Server use the LTPA security token that is previously created to avoid repeated login challenges.

### SPNEGO web authentication in a single Kerberos realm

SPNEGO web authentication is supported in a single Kerberos realm (domain). The challenge-response handshake process is shown in the following figure:

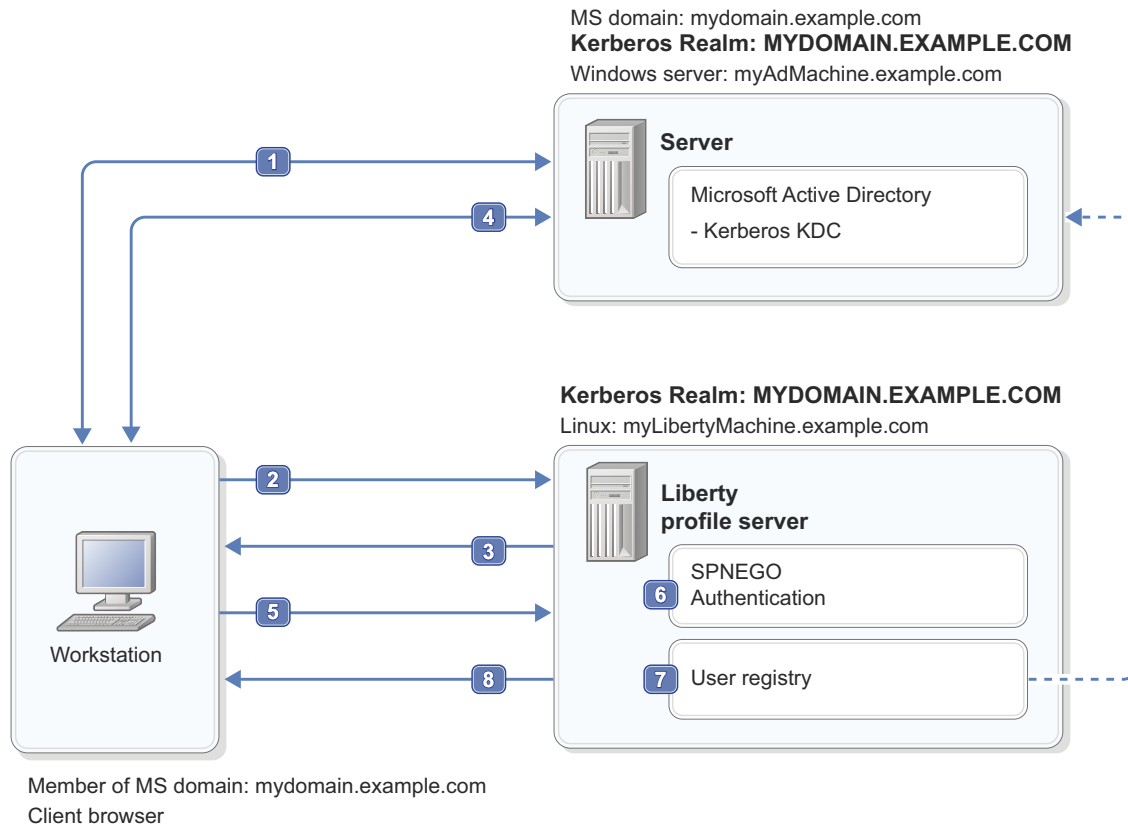


Figure 8. SPNEGO web authentication in a single Kerberos realm

In the previous figure, the following events occur:

1. To begin, the user logs on to the Microsoft domain controller MYDOMAIN.EXAMPLE.COM from the workstation.
2. Next, the user attempts to access the Web application. The user requests a protected Web resource using a client browser, which sends an HTTP GET request to the Liberty server.
3. SPNEGO authentication in the Liberty server answers the client browser with an HTTP 401 challenge header that contains the Authenticate: Negotiate status.
4. The client browser recognizes the negotiate header because the client browser is configured to support integrated Windows authentication. The client parses the requested URL for the host name. The client uses the host name to form the target Kerberos service principal name (SPN) HTTP/myLibertyMachine.example.com to request a Kerberos service ticket from the Kerberos ticket-granting service (TGS) in the Microsoft Kerberos KDC (TGS\_REQ). The TGS then issues a Kerberos service ticket (TGS\_REP) to the client. The Kerberos service ticket (SPNEGO token) proves both the user's identity and permissions to the service (Liberty server).
5. The client browser then responds to the Liberty server Authenticate: Negotiate challenge with the SPNEGO token that is obtained in the previous step in the request HTTP header.
6. SPNEGO authentication in the Liberty server sees the HTTP header with the SPNEGO token, validates the SPNEGO token, and gets the identity (principal) of the user.
7. After the Liberty server gets the identity of the user, it validates the user in its user registry and performs the authorization checks.
8. If access is granted, the Liberty server sends the response with an HTTP 200. The Liberty server also includes an LTPA cookie in the response. This LTPA cookie is used for subsequent requests.

**Note:** Other clients (for example, web services, .NET and J2EE) that support SPNEGO do not have to follow the challenge-response handshake process as shown previously. Those clients can obtain a ticket-granting ticket (TGT) and a Kerberos service ticket for the target server, create an SPNEGO token, insert it in the HTTP header, and then follow the normal process for creating an HTTP request.

### **SPNEGO web authentication in trusted Kerberos realms**

SPNEGO web authentication is also supported in trusted Kerberos realms. The challenge-response handshake process is shown in the following figure:

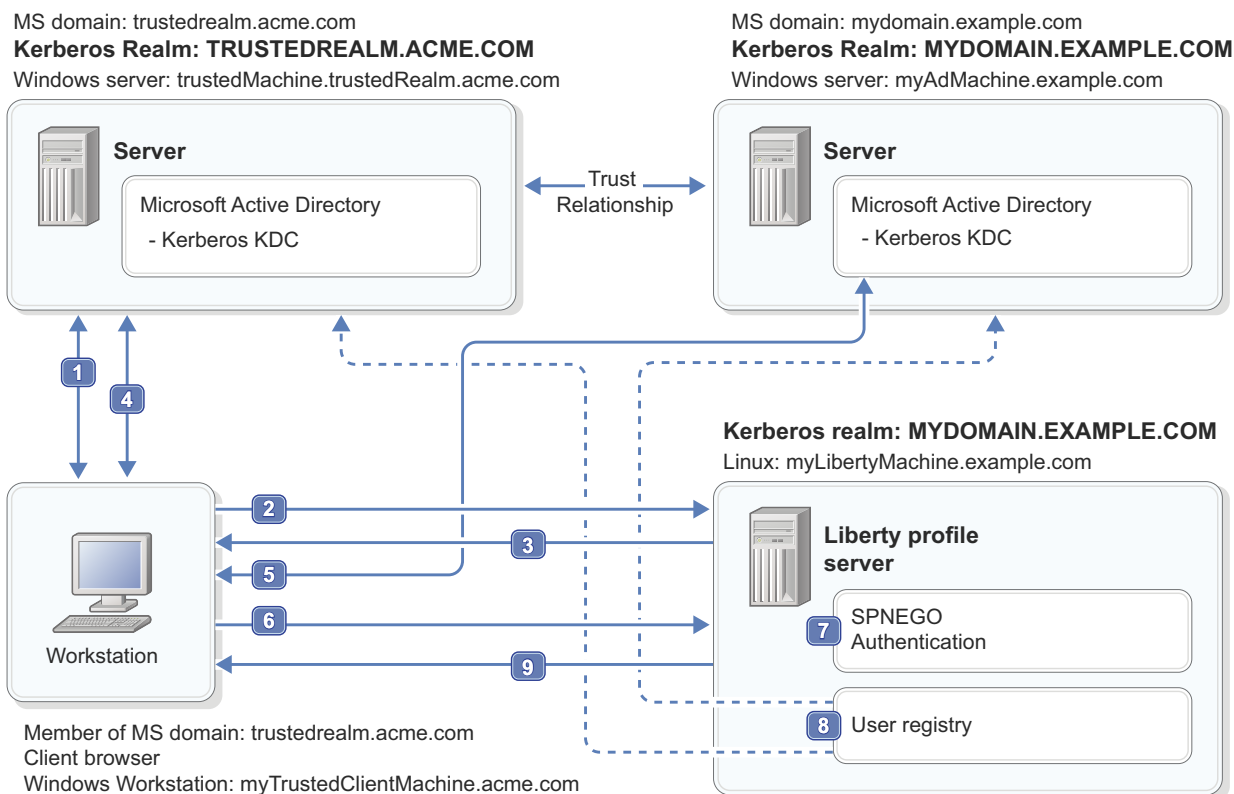


Figure 9. SPNEGO web authentication in a trusted Kerberos realm

In the previous figure, the following events occur:

1. The user logs in to the Microsoft domain controller TRUSTEDREALM.ACME.COM.
2. From a client browser, the user makes a request for a protected Web resource that is hosted on a Liberty server in the original Microsoft domain controller, MYDOMAIN.EXAMPLE.COM.
3. The Liberty server answers the client browser with an HTTP 401 challenge header that contains the `Authenticate: Negotiate` status.
4. The client browser is configured to support integrated Windows authentication. The client browser parses the URL by using the host name of the workstation that hosts the Liberty server application. The client browser uses the host name as an attribute to request a Kerberos cross-realm ticket (TGS\_REQ) for MYDOMAIN.EXAMPLE.COM from realm TRUSTEDREALM.ACME.COM.
5. The client browser uses the Kerberos cross-realm ticket from step 4 to request a Kerberos service ticket from realm MYDOMAIN.EXAMPLE.COM. The Kerberos service ticket (SPNEGO token) proves the user's identity and permissions to the service (Liberty server).
6. The client browser then responds to the Liberty server `Authenticate: Negotiate` challenge with the SPNEGO token that is obtained in the previous step in the request HTTP header.
7. The Liberty server receives the request and checks the HTTP header with the SPNEGO token. It then extracts the Kerberos service ticket, validates the ticket, and gets the identity (principal) of the user.
8. After the Liberty server gets the identity of the user, it validates the user in its user registry and performs the authorization checks.
9. If access is granted, the Liberty server sends the response with an HTTP 200. The Liberty server also includes an LTPA cookie in the response. This LTPA cookie is used for subsequent requests.

**Note:** No modification is required to the Liberty server to support more trusted realms. A trust relationship between the necessary Active Directory realms is the only requirement for SPNEGO to work with trusted realms.

In the trusted Kerberos realms environment, be aware that the Kerberos trusted realm setup has to be completed on each of the Kerberos KDCs. See your Kerberos Administrator and User's guide for more information about how to set up Kerberos trusted realms.


## Support information for SPNEGO web authentication with a browser client

The following scenarios are supported:

- Cross-forest trusts
- Domain trust within the same forest
- Kerberos realm trust

The following scenarios are not supported:

- Forest external trusts
- Domain external trusts

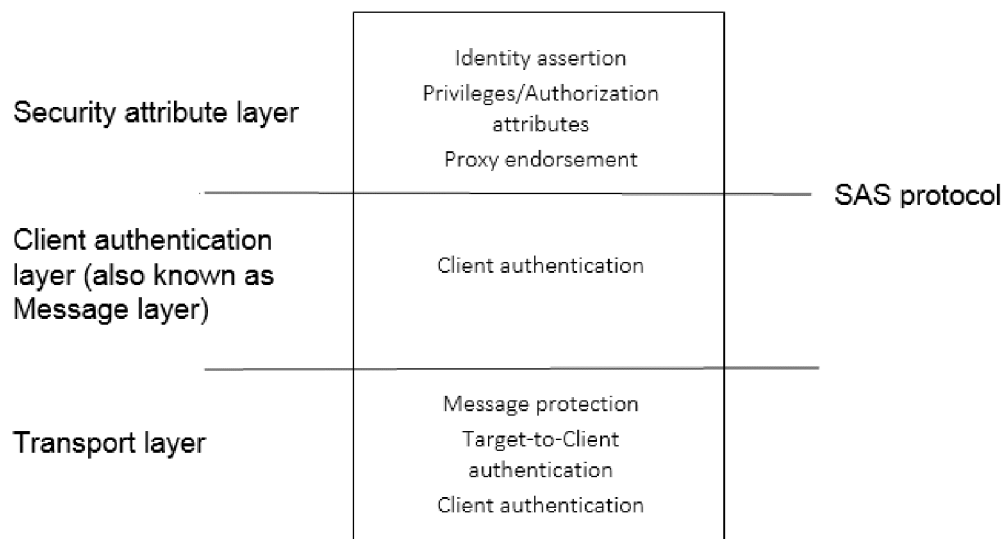
For further information on configuring SPNEGO on the Liberty server, see  **8.5.5.5** Configuring SPNEGO authentication in Liberty.

## Common Secure Interoperability version 2 (CSIv2)

**8.5.5.6**

The Common Secure Interoperability version 2 (CSIv2) is an architecture to satisfy the CORBA security interoperability requirements for authentication, delegation, and privileges. The SAS protocol is used in the CSIv2 architecture to exchange tokens in the service contexts of GIOP request and reply messages for the establishment of security contexts. Transport layer security (SSL/TLS) is required by SAS, and it provides two more layers on top for client authentication and delegation.

The SAS protocol is divided into two layers. The authentication layer is used to perform client authentication where sufficient authentication might not be accomplished in the transport. The attribute layer might be used by a client to push or deliver security attributes, like an identity to a target server where they might be applied in access control decisions. The transport is referred to as another layer in the CSIv2 documentation for ease of discussion, although it is not part of the SAS protocol message and the SAS message sits on top of the transport.





## CSIV2 identity assertion

The CSIV2 identity assertion support in the attribute layer is used to assert an identity from a client process to a server process when the request is performed by using RMI/IIOP.

An assertion is a declaration of one entity to another to accept an identity on its behalf. A client can assert an identity that represents the subject that was effective at the time it started the remote resource. In addition to the identity token that is representing the user, the client process also sends its own identity in either the authentication layer or the transport layer. The target server ensures that the client process is able to assert an identity by performing a trust validation. If the target server trusts the client, then the server uses the asserted identity to create a server-side subject that represents the user that was effective at the client process at the time of invocation.

The client can assert a user who is using a Principal Name identity token. The format of the principal name depends on the user registry that is configured at the client process. The anonymous identity token type is also supported and the server uses the unauthenticated subject when it receives such token.

For information on configuring the CSIV2 attribute layer with identity assertion, see “Configuring inbound CSIV2 attribute layer” on page 1277 or “Configuring outbound CSIV2 attribute layer” on page 1282.

## CSIV2 authentication layer

The CSIV2 authentication layer is used to carry authentication information from a client process to a server process when the request is performed by using RMI/IIOP.

The CSIV2 authentication layer can contain a token that is sent by the client that the server can then use to authenticate the client. Various token types are supported in the authentication layer. For example, the GSSUP token is used to transmit the client's user name and password, which is validated against the user registry of the target server. The Lightweight Third Party Authentication (LTPA) token is a token that represents the client's user without having to transmit a password, but the user must be authenticated at the client process before the remote method invocation and both the client and server processes must share LTPA keys.

With either token type, the token is used to authenticate the remote user at the server process and create a Subject representation of the Subject that was effective at the client side before the client started the remote object. When identity assertion is also enabled, the authentication layer can contain the security information that represents the client identity while the identity assertion token represents the actual remote user at invocation time.

For information on configuring the CSIV2 authentication layer, see “Configuring inbound CSIV2 authentication layer” on page 1279 or “Configuring outbound CSIV2 authentication layer” on page 1284.

## CSIV2 transport layer

The Common Secure Interoperability version 2 (CSIV2) transport layer support is used to protect the SAS protocol request message and support client certificate authentication from a client process to a server process.

The main function of the transport layer is to provide the security characteristics of the transmission of the SAS protocol messages from a client to a server process. The messages can be protected by using encryption, signing, or both. The Liberty SSL support is used as the underlying mechanism that is providing such characteristics.

A second function of the transport layer is to provide a source of authentication material when the authentication layer is not used. If identity assertion is enabled and the authentication layer is not

enabled, then the client process identity is obtained from the transport's client certificate chain. The target server process authenticates the client's certificate chain by mapping it to a user in its user registry. The certificate chain issuer distinguished name is used to determine if the client is trusted to assert an identity.

If no identity assertion and authentication layer are enabled, the subject that is obtained from mapping the client certificate chain is used as the caller subject when the actual remote method call is started at the target server process. This also applies when the target server's authentication layer is supported, but not required, and the client did not send an authentication token and an identity token.

For information on configuring the CSIV2 attribute layer with identity assertion, see [Configuring inbound CSIV2 transport layer](#) or [Configuring outbound CSIV2 transport layer](#).

## Key terms

### ORB – Object Request Broker.

It mediates object method invocations among entities, which might or might not be collocated in the same process.

### Security context

Information that is used to prescribe what the security characteristics are for a particular operation on an object in an ORB. For example, the identity that is to be used during the invocation of the object operation.

### Client security service or CSS

The entity that is initiating a SAS protocol request to establish a security context in the Target Security Service for an operation on an object in the target's ORB.

### Target security service or TSS

The entity that is receiving a SAS protocol request for the establishment of a security context in relationship to an operation on an object in its ORB. It accepts or denies a request to establish or use a security context.

### Client authentication

A token-based mechanism that is used to authenticate the client. GSSUP (Username Password GSS) is the minimum requirement, but there might be others, like LTPA.

### Identity assertion

Mechanism by which an intermediary entity vouches for another entity and the TSS uses the asserted identity for the invocation principal. The TSS can decide whether it trusts the proxy that is asserting the identity or not.

### Stateless

The security context is only used during the duration of a single request and it is not reused for subsequent requests.

### Stateful

The security context can be reused by multiple requests after it is established and until it is invalidated by the TSS or the CSS.

### Transport layer security

The security support that is provided by the underlying transport.

## Authentication on the Liberty application client container



8.5.5.6

The authentication requirements on the client are the same as on the server, but some of the mechanisms to authenticate on the client are different than on the server.

Authentication is required on the client when accessing a protected resource on the server. Follow one of these methods to provide the authentication information:

- Specify a user and password in the `client.xml` file: The credentials are sent to the server using the CSIV2 protocol and it is recommended that you encrypt or encode the password. For further details, see [Configuring the outbound CSIV2 authentication layer in the Liberty application client container](#).

- **Client certificate authentication:** The client presents the server a certificate, which is authenticated and mapped to a user in the registry for authorization checks. To configure the server, see *Configuring inbound CSIV2 transport layer*. To configure the client, see *Configuring the outbound CSIV2 transport layer in the Liberty application client container*.
- **Perform a programmatic login:** Programmatic login is a type of form login that supports application presentation login forms for authentication. This approach requires the application developer to collect the user's credentials and authenticate that user. For further details, see *Configuring a JAAS programmatic login on the Liberty application client container*.

As on the server, you can use a custom login module to either make more authentication decisions or add information to the subject to make finer-grained authorization decisions inside your client application. For further details, see *Configuring a JAAS custom login module for the Liberty application client container*.

## SAML 2.0 Web Browser Single-Sign-On



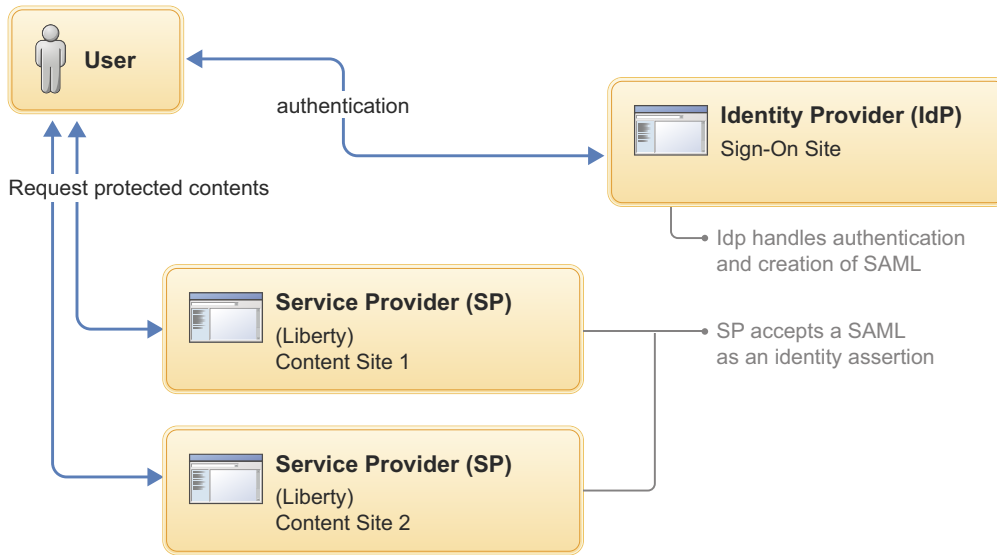
8.5.5.7

SAML Web Browser Single-Sign-On (SSO) enables web applications to delegate user authentication to a SAML identity provider instead of a configured user registry.

Security Assertion Markup Language (SAML) is an OASIS open standard for representing and exchanging user identity, authentication, and attribute information. A SAML assertion is an XML formatted token that is used to transfer user identity and attribute information from the identity provider of a user to a trusted service provider as part of the completion of a single sign-on request. A SAML assertion provides a vendor neutral means of transferring information between federation business partners. Using SAML, an enterprise service provider can contact a separate enterprise identity provider to authenticate users who are trying to access secure content.

The WebSphere Application Server Liberty supports the SAML web browser single sign-on profile with HTTP Post bindings, and acts as a SAML service provider. A web user authenticates to a SAML identity provider, which produces a SAML assertion, and the WebSphere SAML service provider uses the SAML assertion to establish a security context for the web user.

The SAML web SSO flow includes three actors: the end user, the identity provider (IdP), and the service provider (SP). The user always authenticates to the IdP, and the SP relies on IdP assertion to identify the user.

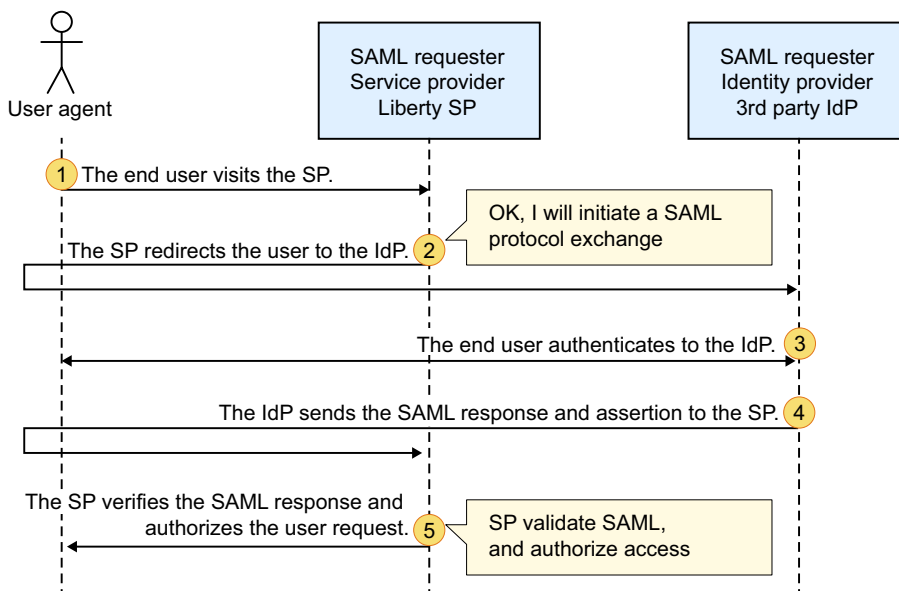


1. User requests a service from SP.
2. SP requests a user identity (SAML) from the IdP via redirect.
3. IdP asks user to login  
(There is no sign-in if user has been authenticated).
4. SP obtains SAML
5. SP makes an access control decision on the basis of SAML assertion

Figure 10. Key Concepts in Web Single-Sign-On

## Liberty SAML Web Browser SSO Scenarios

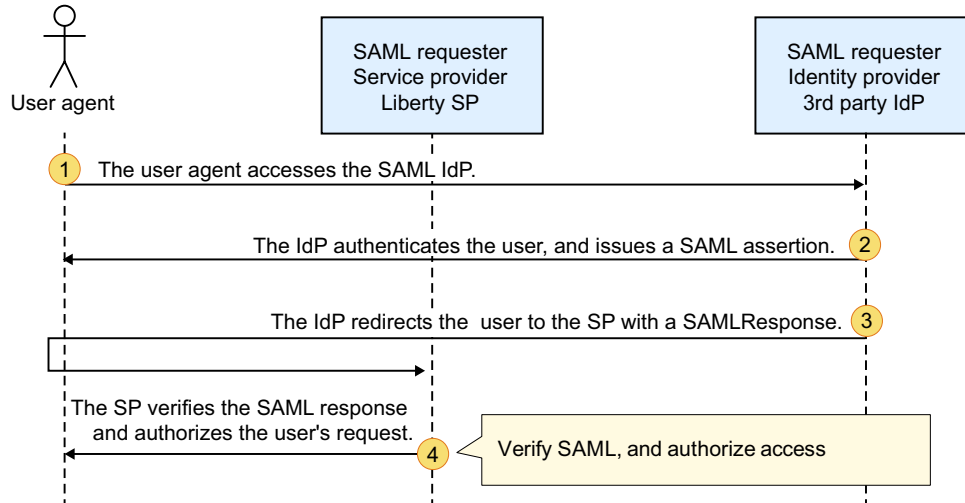
Figure 11. Scenario 1: SP-Initiated solicited Web SSO (End user starting at SP)



1. The end user visits the SP.
2. The SP redirects the user to the IdP.
3. The end user authenticates to the IdP.

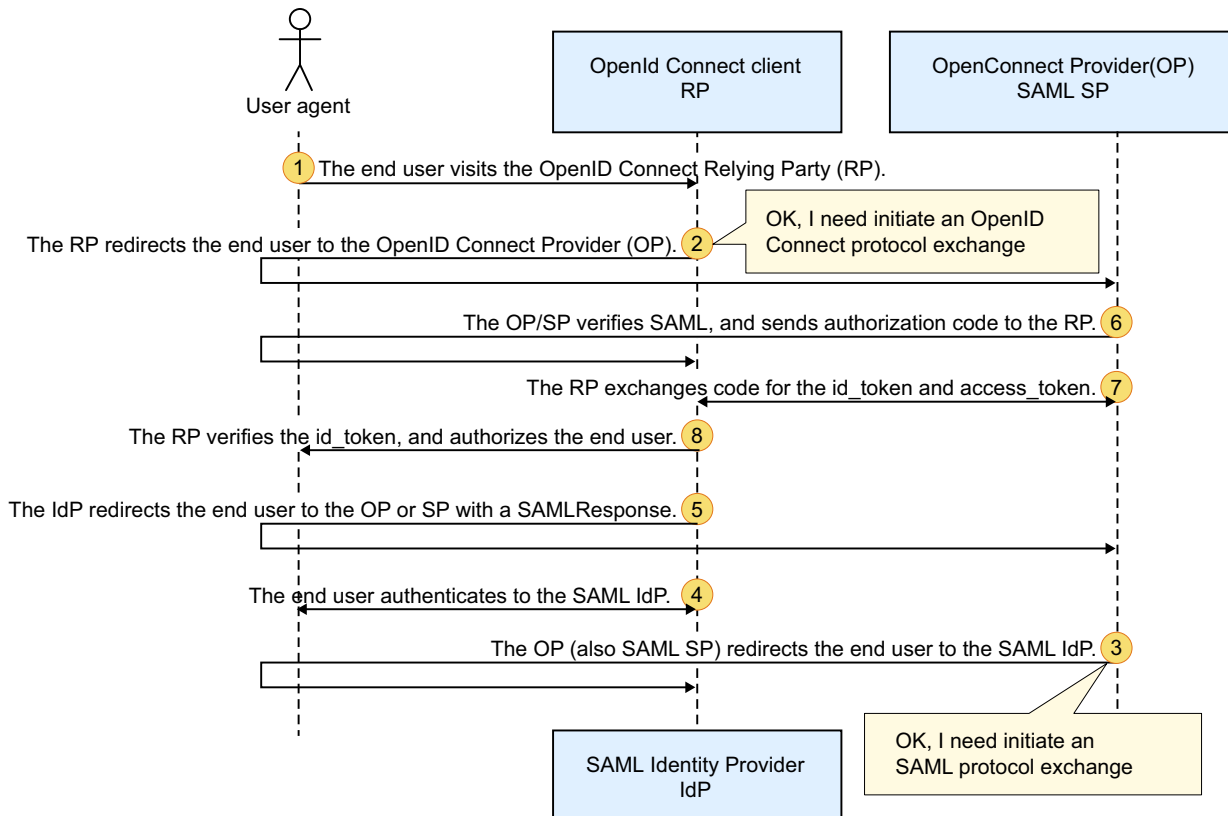
4. The IdP sends the SAML response and assertion to the SP.
5. The SP verifies the SAML response and authorizes the user request.

Figure 12. Scenario 2: IdP-Initiated unsolicited Web SSO (End user starting at IdP)



1. The user agent accesses the SAML IdP.
2. The IdP authenticates the user, and issues a SAML assertion.
3. The IdP redirects the user to the SP with a SAMLResponse
4. The SP verifies the SAML response and authorizes the user's request.

Figure 13. Scenario 3: OpenID Connect provider and SAML service provider



1. The end user visits the OpenID Connect Relying Party (RP).
2. The RP redirects the end user to the OpenID Connect Provider (OP).
3. The OP (also SAML SP) redirects the end user to the SAML IdP.
4. The end user authenticates to the SAML IdP.
5. The IdP redirects the end user to the OP or SP with a SAMLResponse.
6. The OP/SP verifies SAML, and sends authorization code to the RP.
7. The RP exchanges code for the id\_token and access\_token.
8. The RP verifies the id\_token, and authorizes the end user.

## Authorization

Authorization in Liberty determines whether a user has access to a certain role within the system.

Authorization specifies access rights to resources. It usually follows authentication that confirms an identity. Whereas authentication answers the question: “Are you who you say you are?”; authorization answers the question: “Do you have permission to do what you are trying to do?”

The following sections describe these concepts in detail:

- “Authorization for administrative functions”
- “Authorization for applications” on page 607
- “Special subjects” on page 608
- “Access IDs and authorization” on page 608

### Authorization for administrative functions

When an entity attempts to access a resource, the authorization service determines whether that entity has the required rights to access the resource. This concept holds true whether an entity is accessing an

application or performing administrative functions. The main difference between authorizing access to an application and access to an administrative function lies in how the users are mapped to roles. For authorization of applications, use the `application-bnd` element in the `server.xml` file or the `ibm-application-bnd.xml/xmi` file to map the users to roles. For authorization of administrative functions, use the `administrator-role` element in the `server.xml` file to map the users to the administrator role. For more information about administrative security, see “Connecting to Liberty by using JMX” on page 1021.

## Authorization for applications

The following diagram describes how authorization works for applications:

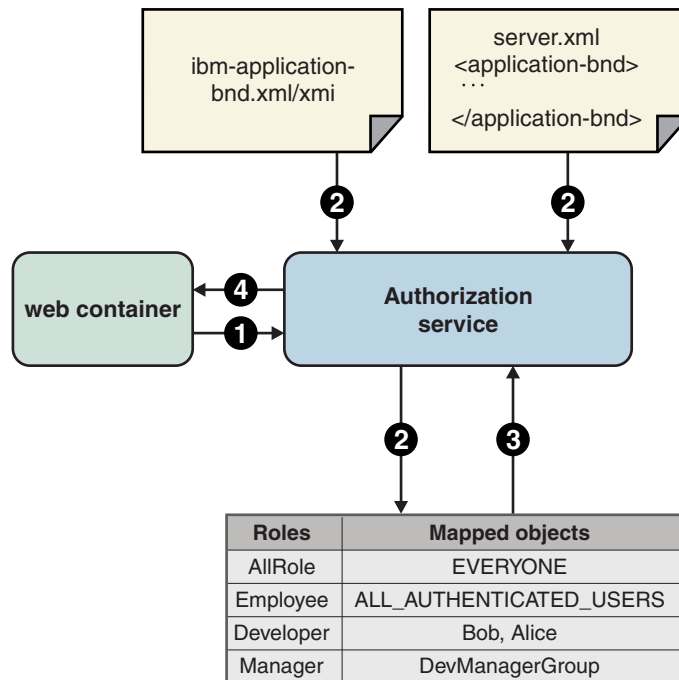


Figure 14. Overview of authorization process

1. An authorization is made when an entity attempts to access a resource in an application that is served by Liberty. The web container calls the authorization service to determine whether a user has permission to access a certain resource, given a set of one or more required roles. The required roles are determined by the `auth-constraint` elements in the deployment descriptor and `@ServletSecurity` annotations.
2. The authorization service determines what objects the required role is mapped to. This step is accomplished by processing the mappings that are defined in the `ibm-application-bnd.xmi` file or the `ibm-application-bnd.xml` file, and the `application-bnd` element of the `server.xml` file. The mappings from these two sources are merged. If the same role is present in both sources, only the role mapping in the `server.xml` file is used. The advantage of using the `server.xml` file for mapping roles to users is that your application does not need to be packaged into an EAR file and it is easier to update. Alternatively, using the `ibm-application-bnd.xml/xmi` file makes your application portable to other servers and other traditional servers that do not support the `server.xml` file.
3. If the required role is mapped to the **EVERYONE** special subject, then the authorization service returns immediately to allow anyone access. If the role is mapped to the **ALL\_AUTHENTICATED\_USERS** special subject and the user is authenticated, then the authorization service grants access to the user. If none of these conditions are met, then the authorization service determines what users and groups are mapped to the required role. The authorization service grants access to the resource if the user is mapped to the required role or if the user is part of a group that is mapped to the role.

- The authorization service returns a result back to the web container to indicate whether the user is granted or denied access.

## Special subjects

When you map entities to roles, you can map a special subject instead of a specific user or group. A special subject is an extension to the concept of a subject. A special subject can represent a group of users that fall under a specific category.

The following two types of special subjects are available:

- **EVERYONE**: represents any entity on the system, which means that no security is available because everyone is allowed access and you are not prompted to enter credentials.
- **ALL\_AUTHENTICATED\_USERS**: represents any entity that successfully authenticates to the server.

To map a special subject to a user, update either the `ibm-application-bnd.xmi/xml` file or the `server.xml` file, where the `application-bnd` is under the `application` element. In this example, the role that is named `AllAuthenticated` is mapped to the special subject `ALL_AUTHENTICATED_USERS`:

```
<application-bnd>
 <security-role name="AllAuthenticated">
 <special-subject type="ALL_AUTHENTICATED_USERS" />
 </security-role>
</application-bnd>
```

See “Configuring authorization for applications in Liberty” on page 1254.

## Access IDs and authorization

When you authorize a user or group, the server needs a way to uniquely identify that user or group. The unique ID of the user and group serve this purpose and are used to build the authorization configuration. These IDs are determined by the user registry implementation: the unique user ID is the value of `getUniqueId()`, and the unique group ID is the value of `getUniqueGroupId()`. You can also choose to explicitly specify an access ID for the user or group in the authorization configuration. These explicit access IDs are used instead of the values that are returned by the user registry implementation. To specify an access ID in the `ibm-application-bnd.xml/xmi` file or the `server.xml` file, where the `application-bnd` is under the `application` element, use the **access-id** attribute for the user or group element.

In this example, an access ID is specified for the user Bob and the group developers:

```
<application-bnd>
 <security-role name="Employee">
 <user name="Bob" access-id="user:MyRealm/Bob"/>
 <group name="developers" access-id="group:myRealm/developers"/>
 </security-role>
</application-bnd>
```

**Note:** The **access-id** attribute is used for the authorization check. If it is not specified, it is determined from the registry that is configured by using the user or group name. However, you must specify the **access-id** attribute as shown in the example when the users or groups do not belong to the active registry. Such as when you are using a programmatic login.

## Security on the Liberty application client container



8.5.5.6

Security on the Liberty application client container includes SSL, JAAS, and CSIV2.



Application clients are client programs that run in their own Java virtual machines. The Liberty application client container provides system services for these clients, including security. The security services on the client are a subset of those that are available on the server.

## Enabling security on the client

To enable security on the client, add the `appSecurityClient-1.0` feature to your `client.xml` file.

```
<featureManager>
 <feature>javaeeClient-7.0</feature>
 <feature>appSecurityClient-1.0</feature>
</featureManager>
```

The `appSecurityClient-1.0` feature enables SSL, CSIV2, and JAAS on the client. You must configure SSL to ensure communications between the client and server are secure and encrypted. For more information, see [Enabling SSL communication for the Liberty application client container](#). CSIV2 provides a protocol for the client to send authentication information to the server. The client in a Liberty application client container is not able to assert identities or propagate security attributes. To understand more about CSIV2 and how to configure it on the client, see [Common Secure Interoperability version 2 \(CSIV2\)](#), and [Configuring Common Secure Interoperability version 2 \(CSIV2\) in the Liberty application client container](#). The JAAS framework on the client enables a client application to gather credentials from the user that is using callbacks, and authenticate that user by using login modules. For more information about authenticating users on the client, see ["Authentication on the Liberty application client container"](#) on page 602.

## Java 2 Security

8.5.5.7

Java 2 Security functionality is supported in WebSphere Application Server Liberty. Java 2 Security provides a policy-based, fine-grained access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources.

Java 2 Security is independent of Java Platform, Enterprise Edition role-based authorization. Java 2 Security guards access to system resources such as file input and output, sockets, and properties; whereas Java Platform, Enterprise Edition security guards access to web resources such as servlets and JSP files.

### Java 2 Security for deployers and administrators

Before you enable Java 2 Security, you need to make sure that all the applications are granted the required permissions, otherwise, applications might fail to run. By default, applications are granted the permissions per the Java Platform, Enterprise Edition 7.0 specification. If an application is not prepared for Java 2 Security or if the application provider does not provide a `permissions.xml` file as part of the application, then the application might cause Java 2 Security access control exceptions at run time when Java 2 Security is enabled. Even if the application is running, it might not run correctly.

### Java 2 Security for application developers

Application developers must understand the permissions that are granted in the default WebSphere policy and the permission requirements of the Java SDK APIs. You need to know whether the APIs that your application calls require extra permissions or not. For more information about which Java APIs require permissions, see [Permissions in the Java 2 SDK](#).

Permissions are added to an application by way of the `permissions.xml` file, and the codebase that is associated with the listed permissions is based on the location of the file. For a stand-alone `.war` application, the `permissions.xml` file is bundled under the `META-INF` directory and all specified

permissions apply to all modules included in the .war file. For a .ear application, the permissions.xml is bundled directly under the META-INF directory for the .ear itself, and the specified permissions apply to all modules included in the .ear file.

**Note:** In the case of an .ear application, permissions.xml files that are bundled under the META-INF directory of any module other than the .ear are ignored.

## Enabling Java 2 Security

Java 2 Security functions are part of the kernel extension and are enabled at bootstrap time by updating the bootstrap.properties file with the websphere.java.security property.

If the websphere.java.security property is specified in the bootstrap.properties file, Java 2 Security is enforced; otherwise, no permission checking occurs.

## Specifying restricted permissions

Liberty provides a mechanism to specify restricted permissions when it runs a web or EJB application component. A restricted permission ensures no instance of that permission is granted to a bundle or application. They provide a mechanism to prevent applications from granting themselves more permissions than what must be allowed, for example, the permission to exit VM.

Restricted permissions are specified in the server.xml and client.xml files. The following example shows how the PropertyPermission that is used to write the System property os.name is restricted. This syntax is identical in both the server.xml and client.xml files:

```
<javaPermission className="java.security.PropertyPermission" name="os.name" actions="write" restriction="true" />
```

## Granting permissions

OSGi bundles can self-regulate the permissions that are granted to the libraries/classes within the bundle through the permissions.perm file.

Applications can also self-regulate permissions that are granted through the permissions.xml file, or by specifying the permission grants in the server.xml and client.xml files.

## OSGi bundle permissions

The OSGi specification provides a mechanism to specify permissions for a bundle through the permissions.perm file in the OSGI-INF directory of the bundle. The mechanism allows fine-grained access control of permissions for the bundle.

The permissions.perm file specifies the maximum permissions that the bundles require.

**Important:** An empty permissions.perm file is not equivalent to no permissions.perm file. Make sure that you have a non-empty permissions.perm file if you want restricted permissions.

## Declaring permissions in the server.xml and client.xml for applications

Permissions without a specified codebase, which are defined in the server.xml and client.xml files apply to all applications on that Liberty server.

You can specify the permissions to be granted in the server.xml and client.xml files as given in the following example. In this example, the PropertyPermission that enables all System properties to be read is granted:

```
<javaPermission className="java.util.PropertyPermission" name="*" actions="read" />
```

You can specify the permissions to be restricted in the `server.xml` and `client.xml` files. The following example shows how the `PropertyPermission` that is used to write the System property `os.name` is restricted. This syntax is identical in both the `server.xml` and `client.xml` files:

```
<javaPermission className="java.security.PropertyPermission" name="os.name" actions="write" restriction="true" />
```

#### Notes:

- A restricted permission has **restriction** set to *true*.
- If an application attempts to grant itself a permission that is defined as a restricted permission, the restricted permission takes precedence over the grant and disallows the grant.

## Declaring permissions in `permissions.xml` for applications

The `permissions.xml` file is a new file that is introduced by the Java EE7 specification. It is packaged under the `META-INF` directory for an application.

For applications packaged as a stand-alone `.war` file, the permissions that are specified at the `META-INF` `WAR` level apply to all modules and libraries that are packaged within the `.war` file.

For applications that are packages in an `.ear` file, the declaration of permissions must be at the `.ear` file level. This permission set is applied to all modules and libraries that are packaged within the `.ear` file or within its contained modules. Any `permissions.xml` file within such packaged modules is ignored, regardless of whether a `permissions.xml` file is supplied for the `.ear` file itself.

For applications that are packaged in a `.rar` file, the declaration of permissions must be at the `META-INF` `RAR` level.

## No-throw option

When Java 2 Security is enabled, the JDK Security Manager throws an `java.security.AccessControl` exception by default when a permission violation occurs. If the exception is not handled, it might cause a runtime failure. To help the developers when they are preparing their applications for Java 2 Security, a `no-throw` option is available. The `no-throw` option allows the `AccessControl` exception to be logged in the `console.log` and `messages.log` but does not cause the application to fail. The `no-throw` option is enabled by specifying `websphere.java.security.norethrow=true` in the `bootstrap.properties` file. The `no-throw` option is not enabled by default, hence you must enable this property by specifying it in the `bootstrap.properties` file.

**Note:** Because this property does not allow the Security Manager to throw the exception, the Security Manager technically does not enforce Java 2 Security. The `no-throw` property must not be used in a production environment.

## Dynamic updates

Dynamic updates to the permission files such as the `permissions.perm`, `permissions.xml`, `server.xml`, and `client.xml` are not supported. Updates to permissions require a Liberty server restart.

## Security public APIs

Security public APIs in Liberty provide a way of extending the security infrastructure.

Liberty contains public APIs that you can use to implement security functions. The security public APIs in Liberty are a subset of the traditional security public APIs. The main classes are `WSSecurityHelper`, `WSSubject`, and `RegistryHelper`. These classes contain a subset of the methods that are available in the WebSphere Application Server traditional versions. There is also a new class `WebSecurityHelper`.

The following sections describe those main classes. There are also other classes such as `UserRegistry`, `WSCredential`, and other exception classes.

All the security public APIs supported by Liberty are in the Java API documentation. The Java API documentation for each Liberty server API is available in a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

### **WSSecurityHelper**

This class contains only the methods `isServerSecurityEnabled()` and `isGlobalSecurityEnabled()`. These calls return true if `appSecurity-2.0` or `zosSecurity-1.0`, among others, is enabled. Otherwise, the methods return false. These methods are carried over from the traditional `WSSecurityHelper` class for compatibility.

#### **Note:**

- There are no cells in Liberty, so there is no distinction in Liberty between global security and server security. Therefore, both methods return the same value.
- The method `revokeSSOCookies(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res)` is not supported in Liberty. Instead, you can use the Servlet 3.0 logout function.
- The method `getLTPACookieFromSSOToken()` is renamed to a new public API class: `WebSecurityHelper`.

### **WSSubject**

This class provides utility methods for querying and setting the security thread context. All methods from the traditional `WSSubject` are supported in Liberty.

**Note:** Java 2 Security is supported but not enabled by default in Liberty. So by default, the Java 2 security checks in `WSSubject` are not performed.

### **RegistryHelper**

This class provides access to the `UserRegistry` object and trusted realm information. In Liberty, it contains the following subset of the traditional methods:

```
public static UserRegistry getUserRegistry(String realmName) throws WSSecurityException
public static List<String> getInboundTrustedRealms(String realmName) throws
WSSecurityException
public static boolean isRealmInboundTrusted(String inboundRealm, String localRealm)
```

**Note:** This method involves dynamic information that could change as OSGI dynamic services change. The values that are retrieved can become stale. The `UserRegistry` references should never be cached.

### **WebSecurityHelper**

This class contains the renamed `getLTPACookieFromSSOToken()` method, which was moved from `WSSecurityHelper`:

```
public static Cookie getSSOCookieFromSSOToken() throws Exception
```

## **Security public API code examples**

The following examples demonstrate how to use security public APIs in Liberty to do a programmatic login and operate on the Subject.

- Example 1: Create a Subject and use it for authorization
- Example 2: Create a Subject and make it as the current Subject on the thread
- Example 3: Get information of the current Subject on the thread

### Example 1: Create a Subject and use it for authorization

This example demonstrates how to use `WSSecurityHelper`, `WSSubject`, and `UserRegistry` to do a programmatic login to create a Java Subject, then perform an action and use that Subject for any authorization that is required.

**Note:** The following code uses `WSSecurityHelper` to check if security is enabled before doing further security processing. This check is used extensively because of the modular nature of Liberty: If security is not enabled, then the security run time is not loaded. `WSSecurityHelper` is always loaded, even if security is not enabled.

```
import java.rmi.RemoteException;
import java.security.PrivilegedAction;

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.CustomRegistryException;
import com.ibm.websphere.security.UserRegistry;
import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
import com.ibm.wsspi.security.registry.RegistryHelper;
public class myServlet {

 ...
 if (WSSecurityHelper.isServerSecurityEnabled()) {
 UserRegistry ur = null;
 try {
 ur = RegistryHelper.getUserRegistry(null);
 } catch (WSSecurityException e1) {
 // record some diagnostic info
 return;
 }
 String userid = "user1";
 String password = "user1password";
 try {
 if (ur.isValidUser(userid)) {
 // create a Subject, authenticating with
 // a userid and password
 CallbackHandler wscbh = new WSCallbackHandlerImpl(userid, password);
 LoginContext ctx;
 ctx = new LoginContext("WSLogin", wscbh);
 ctx.login();
 Subject subject = ctx.getSubject();
 // Perform an action using the Subject for
 // any required authorization
 WSSubject.doAs(subject, action);
 }
 } catch (CustomRegistryException e) {
 // record some diagnostic info
 return;
 } catch (RemoteException e) {
 // record some diagnostic info
 return;
 } catch (LoginException e) {
 // record some diagnostic info
 return;
 }
 }
 ...
 private final PrivilegedAction action = new PrivilegedAction() {
 @Override
 public Object run() {
 // do something useful here
 }
 };
}
```

```

 return null;
 }
};
}

```

### Example 2: Create a Subject and make it the current Subject on the thread

The following example demonstrates how to use `WSSecurityHelper` and `WSSubject` to do a programmatic login to create a Java Subject. Make that Subject the current Subject on the thread, and then restore the original security thread context.

**Note:** The following code uses `WSSecurityHelper` to check if security is enabled before doing further security processing.

```

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
...
if (WSSecurityHelper.isServerSecurityEnabled()) {
 CallbackHandler wscbh = new WSCallbackHandlerImpl("user1", "user1password");
 LoginContext ctx;
 try {
 // create a Subject, authenticating with
 // a userid and password
 ctx = new LoginContext("WSLogin", wscbh);
 ctx.login();
 Subject mySubject = ctx.getSubject();
 Subject oldSubject = null;
 try {
 // Save a ref to the current Subject on the thread
 oldSubject = WSSubject.getRunAsSubject();
 // Make mySubject the current Subject on the thread
 WSSubject.setRunAsSubject(mySubject);
 // Do something useful here. Any authorization
 // required will be performed using mySubject
 } catch (WSSecurityException e) {
 // record some diagnostic info
 return;
 } finally {
 // Put the original Subject back on the thread context
 if (oldSubject != null) {
 try {
 WSSubject.setRunAsSubject(oldSubject);
 } catch (WSSecurityException e) {
 // record some diagnostic info
 }
 }
 }
 } catch (LoginException e) {
 // record some diagnostic info
 return;
 }
}

```

### Example 3: Get information of the current Subject on the thread

The following example demonstrates how to use `WSSecurityHelper`, `WSSubject`, and `WSCredential` to get information about the current Subject on the thread.

**Note:** The following code uses `WSSecurityHelper` to check if security is enabled before doing further security processing.

```

import java.util.ArrayList;
import java.util.Iterator;
import java.util.Set;

import javax.security.auth.Subject;
import javax.security.auth.login.CredentialExpiredException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.CredentialDestroyedException;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.cred.WSCredential;
...
if (WSSecurityHelper.isServerSecurityEnabled()) {
 // Get the caller's subject
 Subject callerSubject;
 try {
 callerSubject = WSSubject.getCallerSubject();
 } catch (WSSecurityException e) {
 // record some diagnostic info
 return;
 }
 WSCredential wsCred = null;
 Set<WSCredential> wsCredentials =
 callerSubject.getPublicCredentials(WSCredential.class);
 Iterator<WSCredential> wsCredentialsIterator = wsCredentials.iterator();
 if (wsCredentialsIterator.hasNext()) {
 wsCred = wsCredentialsIterator.next();
 try {
 // Print out the groups
 ArrayList<String> groups = wsCred.getGroupIds();
 for (String group : groups) {
 System.out.println("Group name: " + group);
 }
 } catch (CredentialExpiredException e) {
 // record some diagnostic info
 return;
 } catch (CredentialDestroyedException e) {
 // record some diagnostic info
 return;
 }
 }
}
}
}

```




## Configuration differences between the traditional and Liberty: security

The configuration differences in the security capability between Liberty and traditional indicates the items that you might need to know during applications migration.

Liberty security supports only a subset of security features in the traditional. Unless the support is explicitly mentioned in Liberty documentation, you must assume that the support is not available yet.

The following security features are not included in Liberty:

- Not all public APIs and SPIs are supported. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `{wlp.install.dir}/dev` directory.
- Horizontal propagation.
- SecurityAdmin MBean support, therefore methods such as clearing the authentication cache are not available.
- Java 2 Connector (J2C) principal mapping modules support.
- Multiple security domain support.

-  **8.5.5.6** CSIV2 security attribute propagation.
-  **8.5.5.6** Kerberos authentication.
-  **8.5.5.6** SPNEGO on non-IBM JDK.
- Security auditing subsystem that is part of the security infrastructure of the server.

In Liberty, you can configure user-to-role mappings and RunAs users in the `application-bnd` element of the `server.xml` file. For a Run-As entry, the password is optional. In the traditional, you can only configure the Run-AS entry in the `ibm-application-bnd.xml/xmi` file. For a Run-As entry, the password is required. See “Configuring authorization for applications in Liberty” on page 1254.

In Liberty, role names can be referenced by the `HttpServletRequest.isUserInRole` and `EJBContext.isCallerInRole` APIs or by elements in the deployment descriptor without first declaring the role names using the `@DeclareRoles` annotation or the `<security-role/>` element in the deployment descriptor. However, roles must be declared before being used in WebSphere Application Server traditional.

## The limits to protection through password encryption

Liberty supports Advanced Encryption Standard (AES) encryption for passwords that are stored in the `server.xml` file. When you use this option for protecting system passwords in the Liberty configuration, you need to understand the limits to the protection it provides.

Encrypting a password in the Liberty configuration does not guarantee that the password is secure or protected; it only means that someone who can see the encrypted password, but does not know the encryption key, cannot easily recover the password. The application server process requires access to both the encrypted password and the decryption key, so both these data items need to be stored on the file system that is accessible to the server runtime environment. The encryption key is also required by anyone who encrypts a password that is placed in the server configuration. For an attacker that has access to exactly the same set of files as the Liberty server instance, applying AES encryption to the password therefore provides no additional security over and above “exclusive or” (XOR) encoding.

Nonetheless, there are still reasons why you might consider encrypting passwords in the Liberty configuration. The Liberty configuration is designed to be highly composable and sharable. The administration subsystem of traditional (the administrative console and `wsadmin` scripting) prevents an administrator from gaining access to an XOR-encoded password. Liberty is designed to be configured without an administration subsystem, and so any XOR-encoded password is visible to any administrator. Given these design features, consider the following scenarios:

- The passwords are not sensitive, so encoding them provides little value.
- The passwords are sensitive, so either the configuration files containing the password are security sensitive and access needs to be controlled, or the passwords are encrypted and the encoding key is then protected as security sensitive.

The encryption key used for decrypting can be overridden from the default by setting the `wlp.password.encryption.key` property. This property should not be set in the `server.xml` file that stores the password, but in a separate configuration file that is included by the `server.xml` file. This separate configuration file should contain only a single property declaration, and should be stored outside the normal configuration directory for the server. This ensures that the file containing the key is not included when you are running the `server dump` or `package` command. The encryption key property can also be specified as a bootstrap property. If you choose this option, you should put the encryption key in a separate `properties` file that is included in the `server bootstrap.properties` file.

For information about using XOR or AES to protect your passwords see the related links, especially “`securityUtility` command” on page 1162.



---

## Java Persistence API (JPA)

8.5.5.6

Data Persistence is a means for an application to persist and retrieve information from a non-volatile storage system. Persistence is vital to enterprise applications because of the required access to relational databases. Applications that are developed for this environment must manage persistence themselves or use third-party solutions to handle database updates and retrievals with persistence. The Java Persistence API (JPA) provides a mechanism for managing persistence and object-relational mapping and functions since the EJB 3.0 specifications.

The JPA specification defines the object-relational mapping internally, rather than relying on vendor-specific mapping implementations. JPA is based on the Java programming model that applies to Java Enterprise Edition (Java EE) environments, but JPA can function within a Java SE environment for testing application functions.

JPA represents a simplification of the persistence programming model. The JPA specification explicitly defines the object-relational mapping, rather than relying on vendor-specific mapping implementations. JPA standardizes the important task of object-relational mapping by using annotations or XML to map objects into one or more tables of a database. To further simplify the persistence programming model:

- The EntityManager API can persist, update, retrieve, or remove objects from a database.
- The EntityManager API and object-relational mapping metadata handle most of the database operations without requiring you to write JDBC or SQL code to maintain persistence.
- JPA provides a query language, extending the independent EJB querying language (also known as JPQL), that you can use to retrieve objects without writing SQL queries specific to the database that you are working with.

JPA is designed to operate both inside and outside of a Java Enterprise Edition (Java EE) container. When you run JPA inside a container, the applications can use the container to manage the persistence context. If there is no container to manage JPA, the application must handle the persistence context management itself. Applications that are designed for container-managed persistence do not require as much code implementation to handle persistence, but these applications cannot be used outside of a container. Applications that manage their own persistence can function in a container environment or a Java SE environment.

Java EE containers that support the EJB 3.x programming model must support a JPA implementation, also called a persistence provider. A JPA persistence provider uses the following elements to enable easier persistence management in an EJB 3.x environment:

### **Persistence unit**

Defines a complete Object-Relational Model mapping Java classes (entities + supporting structures) with a relational database. The EntityManagerFactory uses this data to create a persistence context that can be accessed through the EntityManager.

### **EntityManagerFactory**

Used to create an EntityManager for database interactions. The application server containers typically supply this function, but the EntityManagerFactory is required if you are using JPA application-managed persistence. An instance of an EntityManagerFactory represents a Persistence context.

### **Persistence context**

Defines the set of active instances that the application is manipulating currently. You can create the persistence context manually or through injection.

### **EntityManager**

The resource manager that maintains the active collection of entity objects that are being used by the application. The EntityManager handles the database interaction and metadata for

object-relational mappings. An instance of an EntityManager represents a Persistence context. An application in a container can obtain the EntityManager through injection into the application or by looking it up in the Java component name-space. If the application manages its persistence, the EntityManager is obtained from the EntityManagerFactory.

### Entity objects

A simple Java class that represents a row in a database table in its simplest form. Entities objects can be concrete classes or abstract classes. They maintain states by using properties or fields.

## Java Persistence API (JPA) feature overview



8.5.5.6

There are two JPA features that you can use for your application. `jpa-2.0` is built on the Apache OpenJPA open source project. `jpa-2.1` is built on the EclipseLink open source project.

- `jpa-2.0`
- `jpa-2.1`
- JPA feature compatibility

### `jpa-2.0`

Java Persistence API (JPA) 2.0 for WebSphere Application Server is built on the Apache OpenJPA 2.2.x open source project.

Apache OpenJPA is a compliant implementation of the JPA 1.0 and 2.0 specifications. Using OpenJPA as a base implementation, WebSphere Application Server employs extensions to provide more features and utilities for WebSphere Application Server customers. Because JPA for WebSphere Application Server is built from OpenJPA, all OpenJPA function, extensions, and configurations are unaffected by the WebSphere Application Server extensions. You do not need to make changes to OpenJPA applications to use these applications in WebSphere Application Server.

JPA for WebSphere Application Server provides more than compatibility with OpenJPA. JPA for WebSphere Application Server contains a set of tools for application development and deployment. Other features of JPA for WebSphere Application Server include support for DB2<sup>®</sup> Optim pureQuery Runtime, DB2 optimizations, JPA Access Intent, enhanced tracing capabilities, command scripts, and translated message files. The provider of JPA for this product is `com.ibm.websphere.persistence.PersistenceProviderImpl`.

Apache OpenJPA supports the use of properties to configure the persistent environment. You can specify JPA for WebSphere Application Server properties with either the `openjpa` or `wsjpa` prefix. You can mix the `openjpa` and `wsjpa` prefixes as you want for a common set of properties. Exceptions to the rule are the `wsjpa` specific configuration properties, which use the `wsjpa` prefix. When a JPA for WebSphere Application Server-specific property is used with the `openjpa` prefix, a warning message is logged indicating that the offending property is treated as a `wsjpa` property. The reverse does not hold true for the `openjpa` prefix. In that case, the offending property is ignored.

### `jpa-2.1`

Java Persistence API (JPA) 2.1 for WebSphere Application Server is built on the EclipseLink open source project. EclipseLink is the reference implementation for all version of the JPA specification. The provider of JPA for this product is `org.eclipse.persistence.jpa.PersistenceProvider`.

The JPA 2.1 specification added new features that are not available in the JPA 2.0 specification. These features include:

- Schema generation

- Type conversion methods
- Entity graphs in queries and find operations
- Unsynchronized persistence contexts
- Stored procedure invocation
- Injection into Entity listener classes
- JPQL enhancements
- Criteria API enhancements
- Mapping of native queries

Refer to the JPA 2.1 specification for more details on these features. This product also provides a subset of the EclipseLink APIs. See the Liberty feature page, Java Persistence API 2.1, for details.

**Note:** JPA 2.1 is backwards compatible with JPA 2.0.

## JPA feature compatibility

### jpa-2.0

The `jpa-2.0` feature is the JPA 2.0 specification implementation and is backed by Apache OpenJPA. This feature is a part of the Java Platform, Enterprise Edition (Java EE) 6 family of technologies, but it is special as it is compatible with other Java EE 7 features. For example, the `servlet-3.1` feature, an Java EE 7 feature, is used with the `jpa-2.0` feature. This enables applications to stay with the existing JPA provider, but also use new Java EE 7 features.

### jpa-2.1

The `jpa-2.1` feature is the JPA 2.1 specification implementation and is backed by EclipseLink. This feature is only compatible with other Java EE 7 features. If the `jpa-2.1` feature is used with other Java EE 6 features, the following error is emitted into the `message.log` file.

```
CWWKF0033E: The singleton features com.ibm.websphere.appserver.javaeeCompatible-7.0 and com.ibm.websphere.appserver
```

## Java Persistence API 2.1 behavior changes

8.5.5.6

If you already use the `jpa-2.0` feature for your applications, you are encouraged to continue using the `jpa-2.0` feature for your existing applications to avoid any migration issues. For new applications, you are encouraged to use the `jpa-2.1` feature, which enables you to take advantage of the new features available in the JPA 2.1 specification. If you want to change your existing applications to use the `jpa-2.1` feature instead of the `jpa-2.0` feature, you might need to adjust your application in the migration process.

### Differences between jpa-2.0 and jpa-2.1

There are a few major differences between the `jpa-2.0` and `jpa-2.1` features that you need to be aware of:

#### PersistenceProvider class name

`jpa-2.0`

- IBM provider: `com.ibm.websphere.persistence.PersistenceProviderImpl`
- OpenJPA provider: `org.apache.openjpa.persistence.PersistenceProviderImpl`

`jpa-2.1`

- `org.eclipse.persistence.jpa.PersistenceProvider`

#### Caching behavior

`jpa-2.0`: Caching is disabled by default. If your application needs to take advantage of a L2 cache, you must explicitly enable it.

jpa-2.1: By default, the EclipseLink provider has the L2 cache and QueryCache enabled. You must ensure that this setting is the best option for your applications. If you are running in a distributed environment, like a cluster, you need to disable the cache, or understand that different nodes can have different data.

### Enhancement / weaving differences

jpa-2.0: OpenJPA requires entities to be enhanced. See the documentation on the enhancement of JPA entities for more information.

jpa-2.1: EclipseLink works with unenhanced entities. WebSphere Application Server supports static enhancement.

Some features might be unavailable, such as lazy loading and some performance gains.

- If entities classes are statically enhanced for use with the jpa-2.0 (OpenJPA) provider, the classes must be recompiled before you use the jpa-2.1 provider.

### Data source usage differences

The jpa-2.0 feature sparingly uses the non-jta-datasource, so few non-jta-datasource connections are required when you are tuning an application.

The jpa-2.1 uses a non-jta-datasource connection when reading data and a transaction is not active. This means that non-jta-datasource connection pools need to be larger when you use this feature.

See the OpenJPA -> EclipseLink migration guide page for more differences between the two JPA providers.

## JPA 2.1 features available in OpenJPA

OpenJPA, the JPA 2.0 provider, has features that function similarly to the new JPA 2.1 features. This means that if you have an existing application that uses the jpa-2.0 feature and want to use some of the JPA 2.1 new features, you do not have to switch to the jpa-2.1 feature. Instead, you can use the equivalent of the new feature that is provided by OpenJPA. Some of the key JPA 2.1 features that are available in OpenJPA are:

### Schema Generation

This feature enables you to generate DDL or interact directly with the database to define table schemas based on JPA entity definition. For more information, refer to section 9.4 of the JPA 2.1 Specification.

OpenJPA equivalent feature: Schema Mapper

### Entity Graphs

This feature enables you to specify fetching or processing of a graph of entity objects. For more information, refer to section 3.7 of the JPA 2.1 Specification.

OpenJPA equivalent feature: FetchPlan and FetchGroup

### Stored Procedure Queries

This feature enables you to invoke procedures stored in databases. For more information, refer to section 3.10.17 of the JPA 2.1 Specification.

OpenJPA equivalent feature: Query invocation

### Basic Attribute Type Conversion

This feature enables you to convert between an attribute entity representation and database representation for basic type attributes. For more information, refer to section 3.8 of the JPA 2.1 Specification.

OpenJPA equivalent feature: Externalizer feature

### **@Index and @ForeignKey annotations**

Refer to sections 11.1.19 and 11.1.23 of the JPA 2.1 Specification.

OpenJPA equivalent feature: OpenJPA's @Index and @ForeignKey

### **Unwrap utility methods for EntityManager, Cache**

Refer to sections 3.1.1 and 7.10 of the JPA 2.1 Specification.

OpenJPA equivalent features: EntityManagerImpl.unwrap() and OpenJPAPersistence.cast()

### **Object construction when mapping results from native SQL**

Refer to Section 3.10.16.2.2 of the JPA 2.1 Specification.

OpenJPA equivalent feature: ResultShape object

---

## **Binary logging**

Binary logging is a high performance log and trace facility based on the High Performance Extensible Logging (HPEL) technology in WebSphere Application Server traditional.

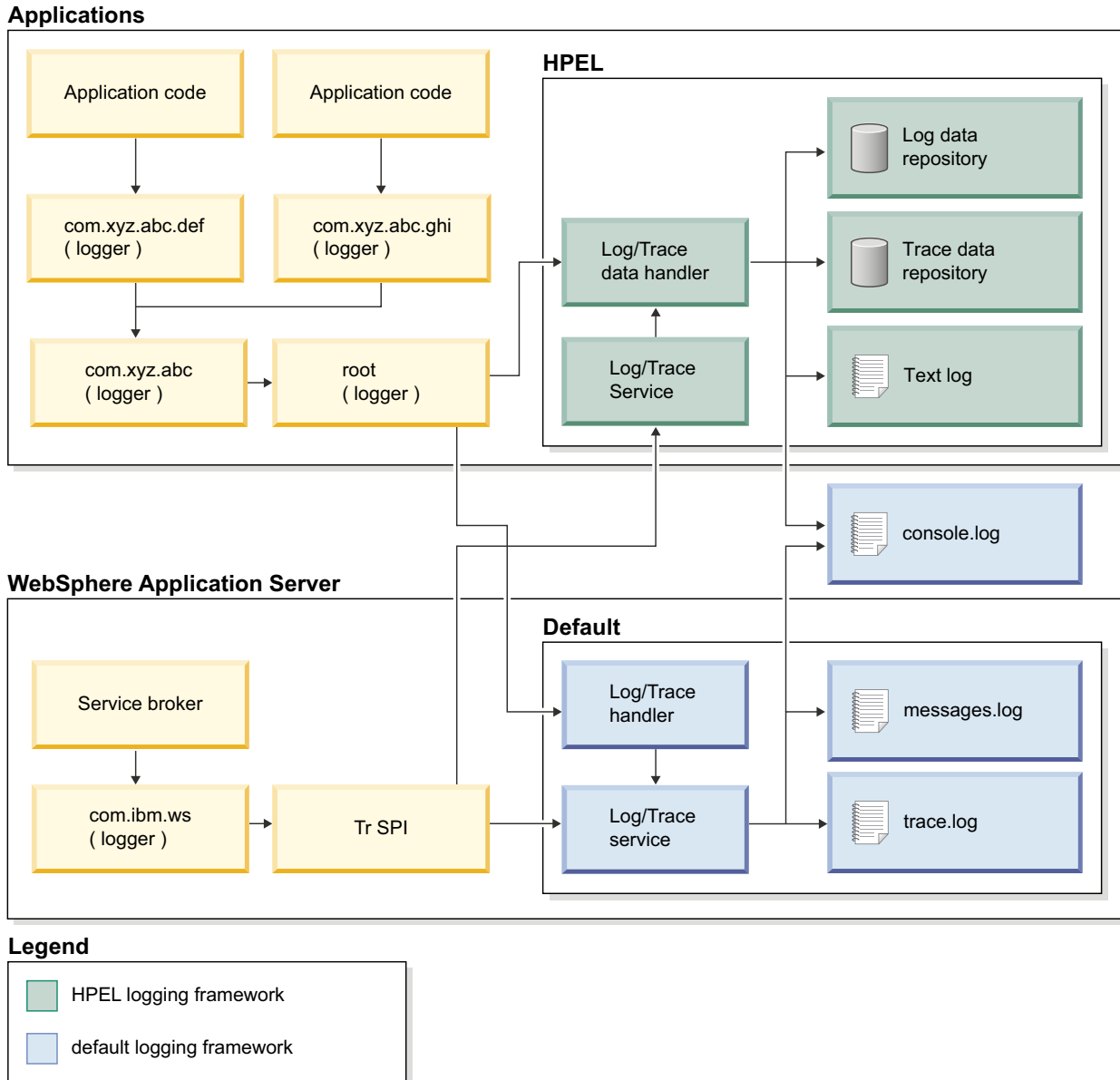
### **Overview**

**Note:** You must enable the binary logging facility to use it.

Binary logging provides a convenient mechanism for storing and accessing log, trace, System.err, and System.out information produced by the application server or your applications. It is an alternative to the default log and trace facility, which provides the JVM logs and diagnostic trace files commonly named messages.log and trace.log.

### **Log and trace storage**

Binary logging provides a log data repository and a trace data repository. See the following figure to understand how applications and the application server store log and trace information.



### Log data repository

The log data repository is a storage facility for log records. Log data is typically intended to be reviewed by administrators. This includes any information applications or the server write to System.out, System.err, OSGi logging service at level LOG\_INFO or higher (including LOG\_INFO, LOG\_WARNING, and LOG\_ERROR), or java.util.logging at level Detail or higher (including Detail, Config, Info, Audit, Warning, Severe, Fatal, and any custom levels at level Detail or higher).

### Trace data repository

The trace data repository is a storage facility for trace records. Trace data is typically intended for use by application programmers or by the WebSphere Application Server support team. This includes any information applications or the server write to the OSGi logging service at level LOG\_DEBUG or java.util.logging at levels below level Detail (including Fine, Finer, Finest, and any custom levels below level Detail).

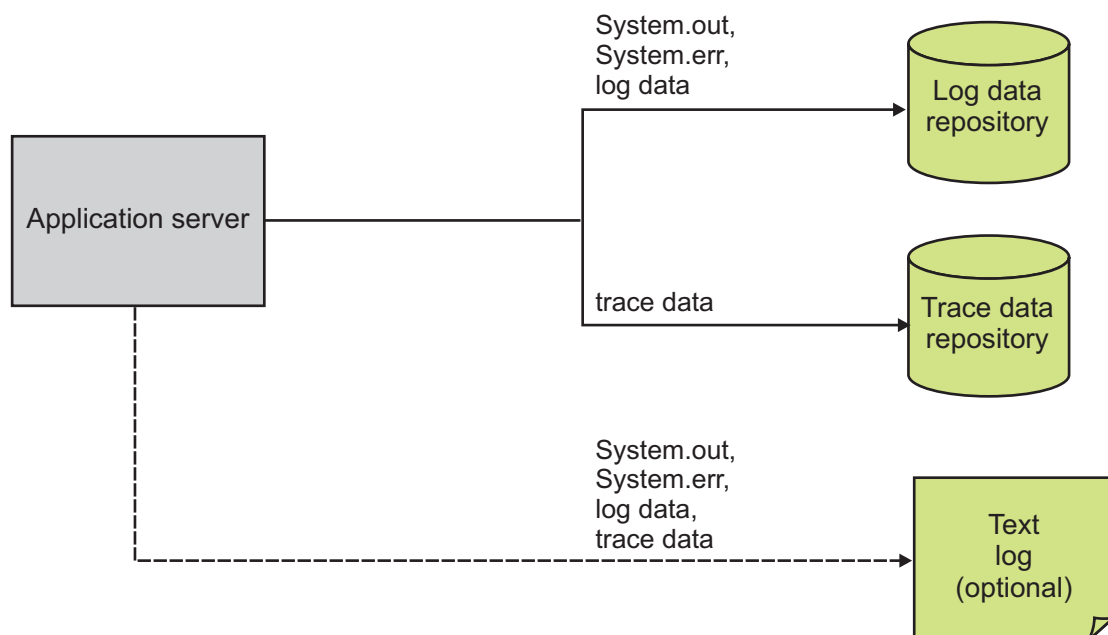
## Log and trace performance

Binary logging has been designed and tested to significantly outperform the default log and trace facility. One result is that the application server can run with trace enabled while causing less impact to performance than tracing the same components using the default log and trace framework. Another result is that applications that frequently write to the logs can run faster when using binary logging.

### Log and trace events are each stored in only one place

Log events, System.out, and System.err are stored in the log data repository. Trace events are stored in the trace data repository. Storing each type of event in only one place ensures that performance is not wasted on redundant data storage.

**Note:** The console log should be disabled in cases where logging performance is important. Any content written to the console log will already be stored in the log data repository.



### Data is not formatted unless it is needed

Formatting data for a user to read uses processor time. Rather than format log event and trace event data at run time, log and trace data are stored more rapidly in a proprietary binary representation. This improves the performance of the log and trace facility. By deferring log and trace formatting until the binaryLog command is run, sections of the log or trace that are never viewed are never formatted.

### Log and trace data are buffered before being written to disk

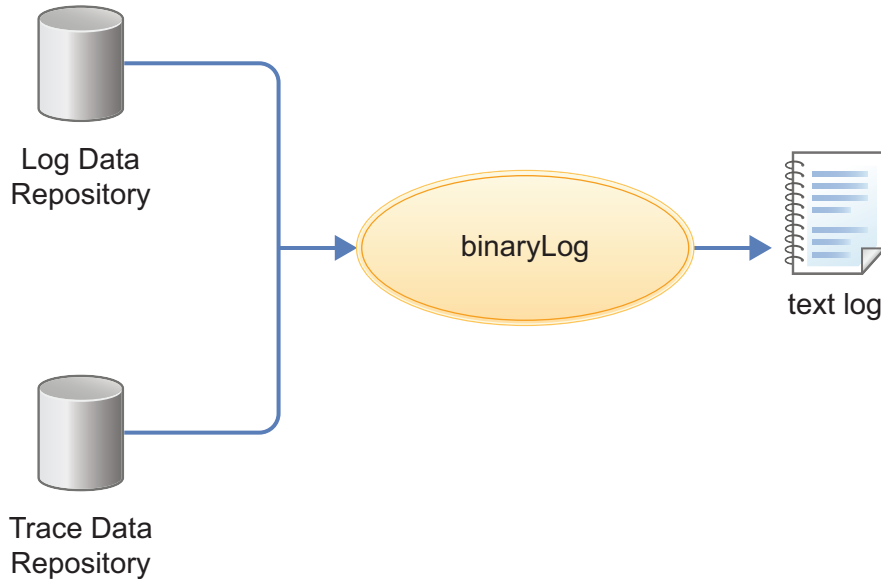
Writing large blocks of data to a disk is more efficient than writing the same amount of data in small blocks. The binary logging facility provides the capability to buffer log and trace data before writing it to disk. By default, log and trace data are stored in an 8 KB buffer before being written to disk. If the buffer is filled within 10 seconds, the buffer is written to disk. If the buffer is not filled within that time it is automatically written to disk to ensure that the logs have the most current information.

## Administration of log and trace

Binary logging has been designed to be easy to configure and understand. For example, administrators can easily configure how much disk space to dedicate to logs or trace, or how long to retain log and trace records, and leave the management of log and trace content up to the server. As another example all log, trace, System.out, and System.err content can be accessed using one easy-to-use command (`binaryLog`), avoiding any possible confusion over which file to access for certain content.

### Reading from the log data and trace data repositories

The log data and trace data repositories are stored in a WebSphere Application Server proprietary format and cannot be read using text file editors such as Notepad or VI. You can copy the log data and trace data repositories in to a plain text format using the `binaryLog` command.



### `binaryLog` command

`binaryLog` is an easy-to-use command-line tool provided for users to work with the log data and trace data repositories. `binaryLog` provides filtering and formatting options that make finding important content in the log data and trace data repositories easy. For example, a user might filter any errors or warnings, then filter all log and trace entries that occurred within 10 seconds of a key error message on the same thread.

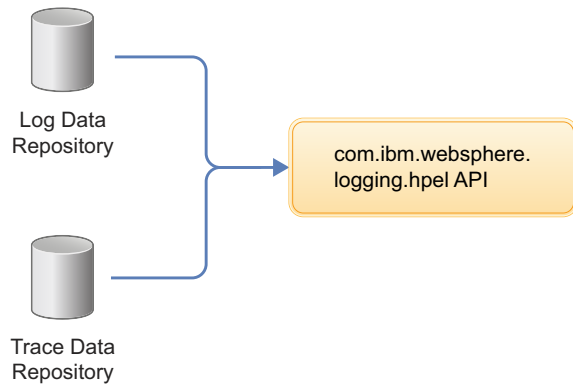
### Filtering using log and trace record extension content

The binary logging facility provides the capability for developers to add custom extensions to log and trace records using a log record context API (`com.ibm.websphere.logging.hpel.LogRecordContext`). You can use the `binaryLog` command-line tool to filter records based on the content of log and trace record extensions.

## Development resources

Binary logging has been designed to make working with log and trace content more flexible and effective than the default logging facility. Log and trace content can be easily filtered to show only the records that are of interest. You can use the command line (see the description of the `binaryLog` command), or developers can create powerful log handling programs using the HPEL API.





## Reading the log data and trace data

An API has been provided to make it easy for developers to develop tools to consume content from the binary log and trace repositories. For example, a developer might write a Java program to search the log and trace content to find any messages with message IDs that match a known list of important message IDs. This API is in the `com.ibm.websphere.logging.hpel` package. Refer to the API documentation for details on the HPEL log reading API.

## Log and trace record extensibility

Developers can add custom extensions to log and trace records through a log record context API (`com.ibm.websphere.logging.hpel.LogRecordContext`). When binary logging stores log and trace records, it includes any extensions present in the log record context on the same thread. For example, a developer might write a servlet filter to add important HTTP request parameters to the log record context. While that servlet runs, HPEL API adds those extensions to any log and trace records created on the same thread.

As with other log and trace record fields, developers can access the record extensions using the HPEL API. This is useful when writing tools to read from log and trace repositories. Developers can also make use of the log record context API to access extensions in custom log handlers, filters, and formatters at run time.

## BinaryLog command options

Use the **binaryLog** command to view or copy the contents of a binary logging repository, or list the available server process instances in the repository. The `binaryLog` command is equivalent to the `logViewer` command in the profile bin directory of the traditional application server.

The binary log and trace facility writes to a repository in a binary format. You can view, query and filter the repository using the `binaryLog` command. The `binaryLog` command provides options for quickly converting repository contents into a text file in various formats, such as basic and advanced formats. The command also provides options to make getting the data you need from the logs easier; for example, allowing you to filter what log records you want by level, logger name, or date and time.

## Syntax

The command syntax is as follows:

```
binaryLog action {serverName | repositoryPath} [options]
```

The value of **options** is different based on the value of **action**.

## Parameters

The following actions are available for the `binaryLog` command:

## view

Read a repository, optionally filter it, and create a version that users can read.

The command syntax is as follows:

```
binaryLog view {serverName | repositoryPath} [options]
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains both the logdata and tracedata directories.

**Note:** If neither a *serverName* nor a *repositoryPath* is specified on the command line, the task is performed against the default server instance, `defaultServer`, if it exists.

Filter options:

All filters are optional. When multiple filters are used, they are logically ANDed together.

- `--minDate=value`  
Filter based on minimum record creation date. Value must be specified as either a date (for example `--minDate="2/20/13"`) or a date and time (for example `--minDate="2/20/13 16:47:21:445 EST"`).
- `--maxDate=value`  
Filter based on maximum record creation date. Value must be specified as either a date (for example `--maxDate="2/20/13"`) or a date and time (for example `--maxDate="2/20/13 16:47:21:445 EST"`).
- `--minLevel=value`  
Filter based on minimum level. Value must be one of `FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`.
- `--maxLevel=value`  
Filter based on maximum level. Value must be one of the following: `FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`.
- `--includeLogger=value[,value]*`  
Include records with specified logger name. Value may include `*` or `?` as a wildcard.
- `--includeMessage=value`  
Filter based on message name. Value may include `*` or `?` as a wildcard.
- `--includeThread=value`  
Include records with specified thread id. Values must be in hexadecimal (for example, `--includeThread=2a`).
- `--includeExtension=name=value[,name=value]*`  
Include records with specified extension name and value. Value may include `*` or `?` as a wildcard. To include a comma in the value, you must use `"\,"`.
- `--includeInstance=value`  
Include records from the specified server instance. Value must either be `"latest"` or be a valid instance ID. Run this command using the `listInstances` action to see a list of valid instance IDs.

Monitor option:

`--monitor`

Continuously monitor the repository and output new content as it is generated.

Output options:

- `--format={basic | advanced | CBE-1.0.1}`

Specify the output format to use. "basic" is the default format.

- `--encoding=value`

Specify the character encoding to use for output.

## copy

Read a repository, optionally filter it, and write the contents to a new repository.

The command syntax is as follows:

```
binaryLog copy {serverName | repositoryPath} targetPath [options]
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains the logdata and tracedata directories.

*targetPath*

Specify the path at which to create a new repository. The *targetPath* must be specified.

**Note:** Either *serverName* or *repositoryPath* must be specified, as well as the *targetPath*.

Filter options:

All filters are optional. When multiple filters are used, they are logically ANDed together.

- `--minDate=value`

Filter based on minimum record creation date. Value must be specified as either a date (for example `--minDate="2/20/13"`) or a date and time (for example `--minDate="2/20/13 16:52:32:808 EST"`).

- `--maxDate=value`

Filter based on maximum record creation date. Value must be specified as either a date (for example `--maxDate="2/20/13"`) or a date and time (for example `--maxDate="2/20/13 16:52:32:808 EST"`).

- `--minLevel=value`

Filter based on minimum level. Value must be one of the following: FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL.

- `--maxLevel=value`

Filter based on maximum level. Value must be one of the following: FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL.

- `--includeLogger=value[,value]*`

Include records with specified logger name. Value may include \* or ? as a wildcard.

- `--excludeLogger=value[,value]*`

Exclude records with specified logger name. Value may include \* or ? as a wildcard.

- `--includeMessage=value`

Filter based on message name. Value may include \* or ? as a wildcard.

- `--includeThread=value`

Include records with specified thread id. Values must be in hexadecimal (for example, `--includeThread=2a`).

- `--includeExtension=name=value[,name=value]*`

Include records with specified extension name and value. Value may include \* or ? as a wildcard. To include a comma in the value, you must use "\",

- `--includeInstance=value`

Include records from the specified server instance. Value must either be "latest" or be a valid instance ID. Run this command using the listInstances action to see a list of valid instance IDs.

## listInstances

List the IDs of server instances in the repository. A server instance is the collection of all log/trace records written from the time a server is started until it is stopped. Server instance IDs can be used with the `--includeInstance` option of the `binaryLog` view action.

The command syntax is as follows:

```
binaryLog listInstances {serverName | repositoryPath}
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains the `logdata` and `tracedata` directories.

**Note:** If *serverName* or *repositoryPath* are not specified on the command line, the task is performed against the default server instance, `defaultServer`, if it exists.

Be aware of `binaryLog` filtering optimizations. The `binaryLog` tool is able to filter log and trace data most efficiently when used with the following filter options:

- `--minDate`
- `--maxDate`
- `--includeThread`
- `--minLevel`
- `--maxLevel`

## Example usage

See the following examples of `binaryLog` commands.

- Display all events in the `defaultServer` repository between July 19th, 2013 and August 2nd, 2013.  

```
binaryLog view --minDate=07/19/13 --maxDate=08/02/13
```
- Display new events from server `myServer`, whose specified level is `WARNING` or higher, using the advanced format as the server writes them to the log repository.  

```
binaryLog view myServer --monitor --minLevel=WARNING --format=advanced
```
- Write log messages from a repository at `/apps/server1/logs`; include only those that were written to the error stream of a specific repository.  

```
binaryLog view /apps/server1/logs --includeLogger=SystemErr
```
- View events from the `defaultServer` repository that occurred before September 14th, 2012 4:28 PM eastern daylight time.  

```
binaryLog view --maxDate="09/14/12 16:28:00:000 EDT"
```
- Write events from the `defaultServer` repository that contain a 'thread' extension with value 'Default Executor-thread-4'  

```
binaryLog view --includeExtension=thread="Default Executor-thread-4" --format=advanced
```
- View the list of server instances in the `defaultServer` repository:  

```
binaryLog listInstances
```

Using `D:\wlp\usr\servers\defaultServer\logs` as repository directory.

Instance ID	Start Date
1358809441761	1/21/13 18:04:01:761 EST
1358864476191	1/22/13 9:21:16:191 EST
1358869523192	1/22/13 10:45:23:192 EST
1358871281166	1/22/13 11:14:41:166 EST
1358879829000	1/22/13 13:37:09:000 EST
1358892222067	1/22/13 17:03:42:067 EST
- View events from the `defaultServer` using one of the instance IDs from the previous example:  

```
binaryLog view --includeInstance=1358871281166
```

- Copy events from the defaultServer, whose specified level is WARNING or higher, from the latest server instance to a new repository at d:\toSupport directory.

```
binaryLog copy defaultServer d:\toSupport --minLevel=warning --includeInstance=latest
```

## Configuring binary logging in Liberty

Use this information as a guide for configuring binary logging in your Liberty.

### About this task

Binary logging provides faster log and trace handling capabilities and more flexible ways to use log and trace content than the default Liberty log and trace framework.

A server configuration consists of a bootstrap.properties file, a server.xml file, and any (optional) files that are included with those files. The bootstrap.properties file specifies properties that need to be available before the main configuration is processed, and are kept to a minimum. The server.xml file is the primary configuration file for the server.

The server.xml file and its associated files use a simple xml format that is suitable for most text editors.

**Distributed operating systems** A richer editing experience is provided by the eclipse server adapter for Liberty (WAS4D+ adapter), which uses a generated schema to provide drop-down lists of available choices, auto-completion, and other editing tools. For a description of the eclipse server adapter for Liberty, see “Editing the Liberty configuration by using developer tools” on page 938.

The bootstrap.properties file specifies whether the server uses binary logging as the log and trace framework, or the default log and trace framework. A server restart is required to switch between binary logging and the default log and trace framework.

You can modify the configuration of binary logging through the server configuration or the bootstrap.properties file.

- Server configuration: To get logging from your own code, which is loaded after server configuration processing, use the server configuration to configure binary logging.
- bootstrap.properties file: You might need to set logging properties to take effect before the server configuration files are processed. For example, if you need to analyze problems that occur early in server start or configuration processing. In this case, you can configure binary logging in the bootstrap.properties file.

You can set Logging properties in either the bootstrap.properties or the server.xml file. Use attributes in the server.xml file, or use equivalent properties in the bootstrap.properties file. Any settings in the bootstrap.properties file are used from the time the server reads the bootstrap.properties file until the time the server.xml file is processed. If the logging properties in the bootstrap.properties file are not replaced or reset in the server.xml file, the property values in the bootstrap.properties file continue to be used.

When binary logging is enabled, the **maxFileSize**, **maxFiles**, **messageFileName**, **traceFileName**, and **traceFormat** logging element attributes are ignored (since binary logging runs without trace.log and messages.log files). The **traceSpecification**, **consoleLogLevel**, and **logDirectory** attributes continue to be used to set the trace specification, the level for the console log, and the placement of the log and trace files.

If you set logging or binary logging attributes in the server.xml file, you can avoid changes in configuration between startup time and runtime by setting the corresponding properties in the bootstrap.properties file to the same value. If no logging or binary logging properties are set in the bootstrap.properties file, the server uses the default logging settings.

## Procedure

- Enable binary logging for the server by updating the `bootstrap.properties` file. In the `bootstrap.properties` file, add the following text on a line by itself:  
`websphere.log.provider=binaryLogging-1.0`
- Use the following parameters to configure binary logging. All subelements that are listed are subelements of the logging element in the `server.xml` file. The following table lists the attributes that are configurable in the `server.xml` file and the equivalent properties that can be set in the `bootstrap.properties` file:

Table 10. Binary logging attributes that are configurable in `server.xml` and the equivalent properties that can be set in `bootstrap.properties`

Logging subelement	Attribute	Equivalent <code>bootstrap.properties</code> property
binaryLog	<code>purgeMaxSize</code>	<code>com.ibm.hpel.log.purgeMaxSize</code>
	<code>purgeMinTime</code>	<code>com.ibm.hpel.log.purgeMinTime</code>
	<code>fileSwitchTime</code>	<code>com.ibm.hpel.log.fileSwitchTime</code>
	<code>bufferingEnabled</code>	<code>com.ibm.hpel.log.bufferingEnabled</code>
	<code>outOfSpaceAction</code>	<code>com.ibm.hpel.log.outOfSpaceAction</code>
binaryTrace	<code>purgeMaxSize</code>	<code>com.ibm.hpel.trace.purgeMaxSize</code>
	<code>purgeMinTime</code>	<code>com.ibm.hpel.trace.purgeMinTime</code>
	<code>fileSwitchTime</code>	<code>com.ibm.hpel.trace.fileSwitchTime</code>
	<code>bufferingEnabled</code>	<code>com.ibm.hpel.trace.bufferingEnabled</code>
	<code>outOfSpaceAction</code>	<code>com.ibm.hpel.trace.outOfSpaceAction</code>

The following example shows a `bootstrap.properties` file that is configured to enable binary logging:  
`websphere.log.provider=binaryLogging-1.0`

The following example shows a `server.xml` file with the binary logging subelements. The log content is set to expire after 96 hours and the trace content is set to retain a maximum of 1024MB:

```
<server description="new server">
 <logging>
 <binaryLog purgeMinTime="96"/>
 <binaryTrace purgeMaxSize="1024"/>
 </logging>
</server>
```

For the full logging configuration reference, see the `logging`, `binaryLog`, and `binaryTrace` elements in the `**** MISSING FILE ****`.

## Results

After you restart the server, binary logging is enabled and configured.

---

## Multimedia

Watch videos, webcasts, and other media about Liberty.

If you do not find the media you want in this topic, see the following websites for media about Liberty and IBM WebSphere Application Server products:

- IBM SupportTV YouTube channel
- WebSphere Application Server YouTube playlists

- IBM Education Assistant for WebSphere Application Server

## Overview

- ▶ Why Liberty? Performance that scales (V8.5.5.6) [Transcript]
- ▶ Why Liberty? Fast application development (V8.5.5.6) [Transcript]
- ▶ Why Liberty? Rapid deployment and powerful administration (V8.5.5.6) [Transcript]
- ▶ Java EE 7 in Liberty (V8.5.5.6) [Transcript]
- ▶ Thoughts on Liberty: Interview with Alasdair Nottingham (V8.5.5) [Transcript]
- ▶ WebSphere Application Server V8.5.5 release overview (V8.5.5)

## Installing and getting started

- ▶ Installing Liberty from a ZIP file (V8.5.5.6) [Transcript]
- ▶ Enabling IHS for Liberty Dynamic Routing (V8.5.5.4) [Transcript]
- ▶ Using the Liberty Repository to enhance Liberty environments (V8.5.5.2) [Transcript]
- ▶ Installing WebSphere eXtreme Scale with Liberty (eXtreme Scale V8.6.0.5) [Transcript]

## Setting up and configuring

- ▶ Configure session cache management with Liberty and WebSphere eXtreme Scale (InterConnect 2015 and eXtreme Scale V8.6.0.6) [Transcript]
- ▶ Setting up Admin Center (V8.5.5.4) [Transcript]

## Administering

- ▶ Touring Admin Center (V8.5.5.7) [Transcript]
- ▶ Getting started with the Server Configuration Tool for WebSphere Liberty (V8.5.5.8) [Transcript]

## Securing

- ▶ OpenID Connect on Liberty (InterConnect 2015) [Transcript]
- ▶ Google OpenID Connect for applications on WebSphere Liberty (InterConnect 2015) [Transcript]

## Developing applications

- ▶ Creating a Hello World Java application (V8.5.5.2) [Transcript]
- ▶ Introduction to WebSphere Application Server Patterns 1.0 [Transcript]

## End-to-end scenarios

- Develop, build, and deploy an application with Liberty server DevOps and some open-source tools.
  - ▶ Demo - DevOps with WebSphere Liberty Server (InterConnect 2015) [Transcript]

## Video: Configure session cache management with Liberty and WebSphere eXtreme Scale

8.5.5.5

The following transcript is for the “Configure session cache management with Liberty and WebSphere eXtreme Scale” video, which describes how to improve the performance of your applications by off loading your current application server and data access processes to an in-memory cache using session cache management with Liberty and WebSphere eXtreme Scale. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content shown in the video.


 Configure session cache management with Liberty and WebSphere eXtreme Scale

Table 11. Introduction. Describe the benefits of session management with Liberty and WebSphere eXtreme Scale.

Scene	Audio	Onscreen Action
1	<p>Caching sessions in your applications is one of the most beneficial and easy-to-do configurations to improve performance and availability of your web applications. If you have an application server, store your information in sessions, then can offload your current process to an in-memory cache to help your applications run at super fast speeds. This video will help you do that by demonstrating how to set up session management quickly and simply with Liberty and WebSphere eXtreme Scale.</p> <p>Liberty is a fast, lightweight and simple Java web application container that application developers can use to develop, test and deploy applications easily.</p>	<p>Show the video title, "HTTP Session Failover for WebSphere Liberty leveraging WebSphere eXtreme Scale".</p>
2	<p>WebSphere eXtreme Scale provides distributed in-memory data storage that is replicated across different machines.</p>	<p>Show IBM Corporation notices and disclaimers.</p>
3	<p>This video includes an overview of session caching with eXtreme Scale. It also includes quick steps to download the Liberty and WebSphere eXtreme Scale for developers. You will also learn how to configure WebSphere eXtreme Scale in Liberty. And finally, a sample web application is included that demonstrates HTTP session failover to an in-memory datagrid that is hosted by WebSphere eXtreme Scale.</p>	<p>List video contents:            Show Quick Demo, Set-Up, and Sample of Liberty with WXS            Overview of Session and Data Caching in WXS            Quick Steps to Download Liberty &amp; WXS            How to Configure WXS in Liberty            Sample Web Application demonstrating HTTP Session Failover to an In-</p>



Table 11. Introduction (continued). Describe the benefits of session management with Liberty and WebSphere eXtreme Scale.

Scene	Audio	Onscreen Action
4	<p>Why is HTTP Session Persistence important? Without HTTP session persistence, you will lose the session data if an application instance fails or becomes unresponsive. For example, in a retail scenario where a user adds items to a shopping cart, that user will most likely have to log in again and rebuild their shopping list if the application fails and session failover is not enabled. Usually that experience creates an unhappy customer.</p> <p>Developers have 3 options to persist HTTP session data:</p> <p>First option: Developers can store sessions locally in the application server memory space, but other application server instances will not share the common user session for web applications. If the instance fails, the session is lost and the user experience is less than desirable.</p> <p>Second option: Developers can persist sessions in a relational database that is shared among instances, but the architecture of relational databases have inherent scalability concerns and reading or writing to a disk is slower than on an in-memory data grid.</p> <p>The third and best option is what this video highlights, where Liberty uses a shared persistence engine to store HTTP session data in the memory of an elastic, scalable architecture, which is known as WebSphere eXtreme Scale. Using Liberty with WebSphere eXtreme Scale allows two or more independent Liberty instances to share a common user session for web applications. When one instance fails, the remaining instances can continue to serve user requests as if no failure had occurred.</p> <p>If maintaining HTTP session high availability is a priority, then use WebSphere eXtreme Scale. Even if a runtime instance fails, the customer session is maintained. The customer is unaware of the failure or maintenance activity, which allows a consistent customer experience without interruption or data loss.</p>	<p>Show information about HTTP session failover: Hands-On Demo: Session Cache HTTP Session Failover for Liberty</p> <p>Customer has an enterprise web application on Liberty, but they w</p>

Table 11. Introduction (continued). Describe the benefits of session management with Liberty and WebSphere eXtreme Scale.

Scene	Audio	Onscreen Action
5	<p>Using WebSphere eXtreme Scale with Liberty offers several benefits to developers:</p> <ul style="list-style-type: none"> <li>• Distributed data access for high availability</li> <li>• In-memory access with rapid retrieval</li> <li>• Scalability built in over traditional methods like relational databases or memory-to-memory session replication</li> <li>• Remote access to eliminate single points of failure</li> <li>• The benefit of a quick configuration in a Liberty environment</li> </ul>	<p>Show information about Liberty data caching: Liberty Data Caching with WXS</p> <p>Supports distributed caching scenarios for web and mobile applications</p> <p>Store key and value objects in memory for fast access.</p> <p>Provides linear scalability, predictable performance, and fault tolerance Replicates data so that cache components may be restarted without data loss</p> <p>Caching is an example of a composable element that can be used to build</p>

Table 12. Demo installing Liberty and WebSphere eXtreme Scale for Developers Liberty. Show commands and server.xml changes.

Scene	Audio	Onscreen Action
6	<p>To experience this capability, you can quickly install Liberty and WebSphere eXtreme Scale for Developers Liberty on your development machine:</p> <ol style="list-style-type: none"> <li>1. Go to WASdev.net.</li> <li>2. Download Liberty.</li> <li>3. Download WebSphere eXtreme Scale for Developers Liberty.</li> <li>4. Install the JAR files into the "hands on" directory, which is the example directory for this demo.</li> </ol>	<p>Show the steps:</p> <ol style="list-style-type: none"> <li>1. Go to the WASdev website.</li> <li>2. Select <b>Want to try the Beta?</b></li> <li>3. From <b>Liberty Repository</b>, select <b>WebSphere eXtreme Scale for Developers Liberty</b>.</li> <li>4. Install the JAR files in my c:\hands-on&gt; directory.</li> </ol>
7	<p>To demonstrate the capabilities of WebSphere eXtreme Scale as a shared, remote HTTP session data grid, you will see a sample application (from the displayed URL) that is deployed locally in the dropin directory of the Liberty web instances, which shows you some examples of placing and retrieving session objects to a data grid.</p>	<p>Show illustrations of the WebSphere eXtreme Scale HTTP session sample and the http://ibm.co/1umQ7iy URL.</p>
8	<p>You will learn how to configure Liberty instances to act as two Liberty web instances that are clients to a WebSphere eXtreme Scale data grid.</p> <p>You will set up a WebSphere eXtreme Scale server as a HTTP session store running on a separate Liberty JVM.</p> <p>And finally, you will configure the two Liberty web instances to store HTTP session data in the data grid of a WebSphere eXtreme Scale container.</p>	<p>Show information about HTTP session failover: Hands-On Demo: Session Cache</p> <p>HTTP Session Failover for Liberty</p> <p>Customer has an enterprise web application on Liberty, but they want</p> <p>A graphic shows ServerA and ServerB in Liberty connected to a session cache.</p>

Table 13. Demo configuring Liberty to connect to the data grid. Show server configuration and testing in a browser.

Scene	Audio	Onscreen Action
9	<p>Now, you will learn to configure Liberty to connect to the data grid. By installing WebSphere eXtreme Scale with Liberty, you get access to features that you can use to manage HTTP session applications that are installed in Liberty.</p> <p>For the remote WXS_Session_Server Liberty instance, the server feature will be set. The server feature contains the capability for running an eXtreme Scale server, which means that both the eXtreme Scale catalog and container are running. Add the server feature when you want to run a catalog server in Liberty or when you want to deploy a data grid application into Liberty.</p> <p>The webGrid feature is to host a session management data grid in the WXS_Session_Server Liberty instance. A Liberty server can host a data grid that caches and then replicates HTTP session data for fault tolerance of applications.</p> <p>Use the webApp feature to enable session management for ServerA and ServerB Liberty instances. The webApp feature contains the capability to extend the Liberty application. Add the webApp feature when you want to replicate HTTP session data for fault tolerance. Remember to set the catalogHostPort to the host and port of the WXS_Session Server Liberty instance.</p>	<p>Show information about configuring a Liberty server.xml to run WXS:</p> <p>The server feature contains the capability for running an eXtreme Scale server, both catalog and container.</p> <pre>&lt;feature&gt;eXtremeScale.server-1.1&lt;/feature&gt; &lt;xsServer isCatalog="true"/&gt;</pre> <p>A Liberty server can host a data grid that caches data for applications to replicate HTTP session data for fault tolerance.</p> <pre>&lt;feature&gt;eXtremeScale.webGrid-1.1&lt;/feature&gt; &lt;xsWebGrid objectGridName="session" catalogHostPort="remoteHost:2600"&gt;</pre> <p>The webApp feature contains the capability to extend the Liberty web application. Add the webApp feature when you want to replicate HTTP session data for fault tolerance.</p> <pre>&lt;feature&gt;eXtremeScale.webApp-1.1&lt;/feature&gt; &lt;httpSession idReuse="true"/&gt; &lt;xsWebGrid objectGridName="session" catalogHostPort="localhost:2600"&gt;</pre>
10	<p>The Session Sample Application is now launched on both ServerA and ServerB instances.</p> <p>Now check that the session attribute in ServerA named Loc is empty.</p> <p>Also check that the same attribute for ServerB is empty as well.</p>	<p>Show a browser open on localhost:9080/HttpSessionWAR/ and a browser open on localhost:9081/HttpSessionWAR/. Both browsers display the WebSphere eXtreme Scale Http Session Sample with no attribute values set.</p> <p>In the browser open on localhost:9080/HttpSessionWAR/, show typing Loc for <b>Attribute</b> in <b>Getting an attribute</b> and clicking <b>Get Attribute</b>. The browser displays <b>Welcome back. Session attribute retrieved.</b> with the Loc attribute set to null. Then show clicking <b>Main Page</b> to return to the previous page.</p> <p>Show these same steps repeated in the browser open on localhost:9081/HttpSessionWAR/.</p>

Table 13. Demo configuring Liberty to connect to the data grid (continued). Show server configuration and testing in a browser.

Scene	Audio	Onscreen Action
11	<p>Now, specify NC as the value for the Loc session attribute in the data grid through Server A's application, and verify that it was set.</p> <p>Now let's go to the ServerB Session Application and retrieve the Session attribute Loc from the data grid.</p>	<p>In the browser open on localhost:9080/ HttpSessionWAR/, show typing Loc for <b>Attribute</b> and NC for <b>Value</b> in <b>Setting an attribute</b> and clicking <b>Set Attribute</b>. The browser displays <b>Welcome back. Session attribute set.</b> with the Loc attribute set to NC. Then show clicking <b>Main Page</b> to return to the previous page. Finally, show clicking <b>Get Attribute</b> and show that the Loc attribute is set to NC, and show clicking <b>Main Page</b> to return to the previous page.</p> <p>Show in the browser open on localhost:9081/ HttpSessionWAR/ clicking <b>Get Attribute</b> and show that the Loc attribute is set to NC. Then show clicking <b>Main Page</b> to return to the previous page.</p>
12	<p>Now you can test this configuration with a simulated, unplanned outage on ServerA by stopping the Liberty instance for ServerA and verifying that ServerB can still retrieve the Loc session attribute from the session cache of the data grid.</p>	<p>Show information about HTTP session failover: Hands-On Demo: Session Cache</p> <p>HTTP Session Failover for Liberty</p> <p>Demonstration of Server failure but Customer Experience is maintained</p> <p>A graphic shows ServerA and ServerB in Liberty. ServerA is crossed out and ServerB is connected to a session cache.</p>
13	<p>ServerA is still running. But the Server A Liberty Instance will be manually stopped using the command line.</p> <p>Now let's go to the browser that was hosting ServerA Session and refresh to show the unavailable Liberty instance.</p>	<p>Show the browser open on localhost:9080/ HttpSessionWAR/. Also show entering a command to stop ServerA at a command line at C:\hands-in\wlp\bin: server stop ServerA</p> <p>Show refreshing the browser open on localhost:9080/HttpSessionWAR/. The message This webpage is not available displays.</p>
14	<p>Now that Server A is down you can verify that ServerB can still detect the Session data in the data grid. To do so, change the Loc session attribute to the value, MD.</p>	<p>Show the browser open on localhost:9081/ HttpSessionWAR/ clicking <b>Get Attribute</b> and show that the Loc attribute is set to NC. Then show clicking <b>Main Page</b> to return to the previous page.</p> <p>Show typing Loc for <b>Attribute</b> and MD for <b>Value</b> in <b>Setting an attribute</b> and clicking <b>Set Attribute</b>. The browser displays <b>Welcome back. Session attribute set.</b> with the Loc attribute set to MD. Then show clicking <b>Main Page</b> to return to the previous page. Finally, show clicking <b>Get Attribute</b> and show that the Loc attribute is set to MD, and show clicking <b>Main Page</b> to return to the previous page.</p>

Table 13. Demo configuring Liberty to connect to the data grid (continued). Show server configuration and testing in a browser.

Scene	Audio	Onscreen Action
15	<p>Now you can simulate ServerA coming back online with the ability to retrieve the new Session value, MD, for attribute Loc, which was just set through ServerB.</p> <p>Now, the browser has been refreshed and points to Server A, where the value for the Loc attribute from the data grid is retrieved.</p>	<p>Show the browser open on localhost:9081/HttpSessionWAR/. Also show entering a command to start ServerA at a command line at C:\hands-in\wlp\bin:</p> <pre>server start ServerA</pre> <p>Show refreshing the browser open on localhost:9080/HttpSessionWAR/. The browser displays the WebSphere eXtreme Scale Http Session Sample with no attribute values set.</p> <p>Finally, show clicking <b>Get Attribute</b> for the Loc attribute to show that it is set to MD, and show clicking <b>Main Page</b> to return to the previous page.</p>


Table 14. Conclusion. Summarize the video content and point viewers to more information.

Scene	Audio	Onscreen Action
16	<p>Congratulations. You've just created your first Liberty cluster with two instances pointing to a shared in-memory data grid for HTTP session storage.</p>	<p>Show information about HTTP session failover:</p> <p>Hands-On Demo: Session Cache</p> <p>HTTP Session Failover for Liberty</p> <p>Demonstration of Server Recovery or additional Liberty Instances</p> <p>A graphic shows ServerA and ServerB in Liberty with both servers connected to a session cache.</p>
17	<p>In this video, you learned the benefits of using a cache for session persistence.</p> <p>No code changes are need to the application to leverage the WebSphere eXtreme Scale data grid.</p> <p>Data for each session persists even with a server outage.</p> <p>And WebSphere eXtreme Scale is scalable up to terabytes of data and can replicate data to thousands of nodes for fault tolerance and high availability.</p>	<p>Show information about Liberty session management:</p> <p>Liberty Session Management with WXS</p> <p>Special purpose elastic in memory cache for storing HTTP session</p> <p>Stores and persists HTTP session objects to the data grid so that</p> <p>No code change to applications using the J2EE standard HTTP sessi</p> <p>Data for each HTTP session survives on a server outage for an app</p> <p>Requires no developer effort to manage</p> <p>Replicates the session data to avoid a single point of failure.</p> <p>Provides low latency data access, transactional semantics.</p>
18	<p>Visit these resources to download and install WebSphere eXtreme Scale on Liberty and to access the sample application that was used in this demo.</p> <p>Thank you for watching and this concludes the video for configuring session cache management with Liberty and WebSphere eXtreme Scale.</p>	<p>Shows resources:</p> <p>(WASdev) <a href="https://developer.ibm.com/wasdev">https://developer.ibm.com/wasdev</a></p> <p>(How to install WXS with Liberty) <a href="http://youtu.be/Zu4Z1GLjM1E">http://youtu.be/Zu4Z1GLjM1E</a></p> <p>(Sample Application used in Demo) <a href="http://ibm.co/1umQ7iy">http://ibm.co/1umQ7iy</a></p>

## Video: DevOps with WebSphere Liberty Server

8.5.5.5

The following transcript is for the “DevOps with WebSphere Liberty Server” video, which demonstrates how to develop, build and deploy an application with Liberty server DevOps and some open source tools. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

 Demo - DevOps with WebSphere Liberty Server

*Table 15. Title page.* Show title and then a basic overview of building, updating, testing, and deploying an application.

Scene	Audio	Onscreen Action
1	This demo will show you how to provide the capability for building and deploying your application with WebSphere Liberty Server and a set of common open source tools.	Show title <i>Demo - DevOps with WebSphere Liberty Server</i> .
2	You will use Git for source control, use WebSphere Application Server Developer Tools to develop your application, and use Maven for build.	Show image of <i>1. Develop and Build Application</i> that uses the following options <ul style="list-style-type: none"> <li>• Git</li> <li>• WebSphere Developer Tools (WDT)</li> <li>• Maven</li> </ul>
3	You will also make changes to your web application running in Liberty and push changes from Eclipse IDE to production environment.	Show image of <i>2. Update Application</i> that uses WDT.
4	You will use Apache Maven together with the WebSphere Liberty Maven plug-in to run integration test to verify your changes.	Show image of <i>3. Test Application</i> with the Liberty Maven plug-in.
5	You will also use Jenkins for continuous integration.	Show image of <i>4. Continuous Integration</i> that uses Jenkins.
6	You will use Chef to push a new build to the production environment.  Liberty will be the Java Platform, Enterprise Edition run time for all the above DevOps scenarios. We will use an Airline application - AcmeAir as a sample in this demo. It will demonstrate a controllable and highly repeatable DevOps practice.	Show image of <i>5. Continuous Deploy</i> that uses Chef.

Table 16. Demo of Developing and Building an Application

Scene	Audio	Onscreen Action
7	<p>Now lets start the first scenario: develop and build your application.</p> <p>Open Eclipse, which has WDT already preinstalled.</p> <ol style="list-style-type: none"> <li>1. We will set the workspace preference for use with Maven.</li> <li>2. Then we will clone a Git repository containing the AcmeAir application, specify the Git repository URI.</li> <li>3. Then we will import the set of Maven based AcmeAir projects: There are 5 Maven-based projects in this repository.</li> <li>4. Now we can run the Maven build to build the projects and create application artifacts. Make sure the build is successful when it finishes.</li> <li>5. Now we can create a Liberty server to run our application from Eclipse: So specify the Liberty install directory.</li> <li>6. And also name the Liberty server.</li> <li>7. Then we can open the server.xml to config the server to run our AcmeAir application. <ul style="list-style-type: none"> <li>• First we will specify the &lt;httpEndpoint&gt; endpoint.</li> <li>• And we will also add the database configuration, which refers to a Derby database for application data.</li> </ul> </li> <li>8. Start the WebSphere Liberty server and the AcmeAir application.</li> <li>9. Now login to the application and experience it.</li> <li>10. You can select your destination and the city from which you depart. Check those available airline tickets and book it.</li> </ol>	<p>Show title <i>Step 1 Develop and Build Application</i>.</p> <p>Demo showing the following steps to develop and build an application that uses WDT.</p> <p>Open Eclipse, which has WDT preinstalled.</p> <ol style="list-style-type: none"> <li>1. Set the workspace preferences for use with Maven. Click the <b>Set all Maven values</b> button.</li> <li>2. We will clone a Git repository containing the AcmeAir application by specify the Git repository URI.</li> <li>3. Import the set of Maven-based AcmeAir projects: The acmeair repository contains 5 Maven-based projects.</li> <li>4. Now run Maven to build the projects and create the application artifacts. Make sure that the build is successful when it finishes.</li> <li>5. Now we can create a WebSphere Liberty server to run the application from Eclipse: Specify the Liberty install directory.</li> <li>6. Name the server AcmeAirDemo and add the acmeair-webapp application into it.</li> <li>7. Open the server configuration server.xml file to configure the Liberty server for service. <ul style="list-style-type: none"> <li>• Replace the &lt;httpEndpoint&gt; element and add further configuration.</li> <li>• Add the database configuration, which refers to a Derby database for application data.</li> </ul> </li> <li>8. Start the WebSphere Liberty server and AcmeAir application.</li> <li>9. Click the <b>application URL</b> to verify that the application is active. Initialize the application by loading the flight data set.</li> <li>10. Now AcmeAir is ready for use. <ul style="list-style-type: none"> <li>• Logins to the application and click the <b>Flights</b> action and type in New York under <b>Leave From</b> and Paris under <b>Arrive At</b>.</li> <li>• Click the <b>Find Flights</b> or <b>Browse Flights</b> button.</li> <li>• Pick any outbound flight and select it by clicking it.</li> <li>• Pick any return flight and select it by clicking it.</li> <li>• Click the <b>Book Select Flights</b> button.</li> </ul> </li> </ol>

Table 17. Demo updating application

Scene	Audio	Onscreen Action
8	In scenario 2 we will make a quick change to our application and show how to do a continuous delivery with that. To simplify the scenario we will just make a very small change to our indexed page. To replace some highlighted content with ""Welcome to Acme Air"". Now after you save these changes, you can refresh to see that the change is in effect.	Show title <i>Step 2 Update Application</i>  Demo showing the Index page that is being updated in the application.

Table 18. Demo testing the application

Scene	Audio	Onscreen Action
9	<p>Next we will use Maven and the Liberty Maven plug-in for integration tests.</p> <ol style="list-style-type: none"> <li>1. The AcmeAir application has a acmeair-itests project that contains integration tests.</li> <li>2. To enable of the integration tests we will need to edit our pom.xml file. There are a few changes we need to make here, but the most important one is that we will enable and config the Liberty Maven plug-in to start the Liberty server before our integration tests and to stop the server after the tests are done.</li> <li>3. We will create a run Configuration for AcmeAir integration tests. <ul style="list-style-type: none"> <li>• Click the <b>Run</b> button to run the integration tests. It will take a few minutes to allow the tests to complete. So its build successfully.</li> <li>• Examine the console output to see the Liberty Maven plug-in starting the server before the tests run and to stop it after the test.</li> </ul> </li> <li>4. You can view the test report and make sure there is no error in the result.</li> <li>5. Next we will push the pom.xml change to Git so that the itests can be run in a Jenkins build.</li> </ol>	<p>Demo testing the application, that uses the following steps</p> <ol style="list-style-type: none"> <li>1. The AcmeAir application has a acmeair-itests project that contains integration tests.</li> <li>2. Enable activation of integration tests by editing the acmeair-itests/pom.xml file.</li> <li>3. Enable the Build Helper Maven plug-in to find an available network port.</li> <li>4. Enable and configure the Liberty Maven plug-in to start the Liberty server before integration tests are executed and stop the server after the tests are done.</li> <li>5. Enable and configure the Maven Failsafe plug-in to execute the integration tests. The tests exercise the REST API of the application.</li> <li>6. Create a run Configuration for AcmeAir integration tests. <ul style="list-style-type: none"> <li>• Click the <b>Run</b> button to run the integration tests. The tests should complete successfully.</li> <li>• Examine the console output to see the Liberty Maven plug-in starting the server before the tests run.</li> <li>• Examine the console output to see the Liberty Maven plug-in stopping the server after the tests.</li> </ul> </li> <li>7. View the test reports under target/failsafe-reports/ directory.</li> <li>8. Push the pom.xml change to Git so the itests can be run in a Jenkins build.</li> </ol>



Table 19. Demo showing continuous integration with Jenkins

Scene	Audio	Onscreen Action
10	<ol style="list-style-type: none"> <li>1. In the next scenario we will use Jenkins to create and run a build job for the AcmeAir application. The job will check out the application code from Git, compile the application, execute the tests, and publish the build artifacts. With the configuration in Schedule, the job will poll the Git repository every two minutes and automatically start a build if any changes are detected. <ul style="list-style-type: none"> <li>• Specify the Liberty install directory.</li> <li>• We will also add a <b>post-build action</b> and choose <b>Archive the artifacts</b>.</li> <li>• After a minute or 2 the build should automatically start.</li> </ul> </li> <li>2. Once the build completes and is successful, examine the build job results. The <b>Test Result</b> should indicate there is no failure.</li> <li>3. Switch back to Eclipse and commit the changes made on the Index page.</li> <li>4. Switch back to Jenkins and you will see within 2 minutes a new build should appear under the <b>Build History</b> table.</li> </ol>	<p>Show title <i>Step 4 Continuous Integration</i></p> <p>Demo showing continuous integration with Jenkins using the following steps:</p> <ol style="list-style-type: none"> <li>1. We will use Jenkins to create and run a build job for the AcmeAir application. The job will check out the application code from Git, compile the application, execute the tests, and publish the build artifacts. With the configuration in Schedule, the job will poll the Git repository every two minutes and automatically start a build if any changes are detected. <ul style="list-style-type: none"> <li>• We will click the <b>Add post-build actions</b> button and choose <b>Archive the artifacts</b>.</li> <li>• After a minute or 2 a build should automatically start.</li> </ul> </li> <li>2. Once the build completes and is successful, examine the build job results. The acmeair-webapp-1.0-SNAPSHOT.war file should appear under <b>Build Artifacts</b> and <b>Test Result</b> should indicate no failures.</li> <li>3. Switch back to Eclipse and commit the changes that are made on the Index page.</li> <li>4. Switch to Firefox and go to http://server:9080/jenkins/job/AcmeAir tests/ address. Within 2 minutes a new build should appear under the <b>Build History</b> table. <ul style="list-style-type: none"> <li>• Click the new build and verify that the right commit message appears under <b>Changes</b>.</li> </ul> </li> </ol>

Table 20. Demo showing continuous deployment with Chef

Scene	Audio	Onscreen Action
11	<p>1. In the last scenario we will do a continuous deployment with Chef.</p> <ul style="list-style-type: none"> <li>• Open a terminal window.</li> <li>• Create an empty cookbook</li> <li>• Edit the metadata file of the cookbook</li> <li>• Edit the recipe file</li> <li>• Upload the AcmeAir cookbook to the Chef server</li> </ul> <p>2. Also register the template node with the Chef server.</p> <p>3. You can config the template node configuration on the Chef server.</p> <p>4. Then you can populate the Chef node. This will cause the <b>chef-client</b> command to run on the template node. The AcmeAir application should be fully deployed and running on the template node. To verify that, switch to Firefox and open the template application server to check the results.</p> <p>5. Then we will create a new Jenkins build job to invoke <b>chef-client</b> on the template node to update it, only after all the integration tests pass.</p> <ul style="list-style-type: none"> <li>• On the Jenkins console, edit the AcmeAir tests job.</li> <li>• Under <b>Post-build Actions</b> click on <b>Add post-build action</b> and choose <b>Build other projects</b>.</li> <li>• Under <b>Projects to build</b> type in AcmeAir-Chef.</li> <li>• Verify that the AcmeAir tests job triggers the AcmeAir-Chef job by requesting a AcmeAir tests build.</li> </ul> <p>We just demonstrate DevOps practice with Liberty and common open source tools. You can now try the full End to End DevOps scenarios all together repeatedly. While in this demo we choose Chef, WDT, Maven, and Jenkins, these technologies can easily be interchangeable with other DevOps tools like uDeploy®, Puppet, etc,, Thanks for watching.</p>	<p>Show title <i>Step 5 Continuous Deploy</i></p> <p>Demo showing continuous deployment with Chef that uses the following steps:</p> <ol style="list-style-type: none"> <li>1. Create a cookbook to deploy AcmeAir <ul style="list-style-type: none"> <li>• Open a terminal window.</li> <li>• Create an empty cookbook</li> <li>• Edit the metadata file of the cookbook</li> <li>• Edit the default.rb recipe file</li> <li>• Upload the AcmeAir cookbook to the Chef server</li> </ul> </li> <li>2. Bootstrap the Chef node and register the template node with the Chef server.</li> <li>3. Configure the template node configuration on the Chef server.</li> <li>4. Populate the Chef node. Execute the <b>knife ssh name:template sudo chef-client</b> command. This will cause the <b>chef-client</b> command to run on the template node. <b>chef-client</b> will execute a set of cookbooks on the node. The AcmeAir application should be fully deployed and running on the template node. To verify, switch to Firefox and open the http://template:9081/acmeair/ address.</li> <li>5. Populate the Chef node automatically with Jenkins. We will create a new Jenkins build job that will invoke <b>chef-client</b> on the template node to update it, only after all the integration tests pass. <ul style="list-style-type: none"> <li>• On the Jenkins console, edit the AcmeAir tests job.</li> <li>• Under <b>Post-build Actions</b> click <b>Add post-build action</b> and choose <b>Build other projects</b>.</li> <li>• Under <b>Projects to build</b> type in AcmeAir-Chef.</li> <li>• Verify that the AcmeAir tests job triggers the AcmeAir-Chef job by requesting a new AcmeAir tests build.</li> </ul> </li> </ol>

## Video: Enabling IHS for Liberty Dynamic Routing

8.5.5.5

The following transcript is for the “Enabling IHS for Liberty Dynamic Routing” video, which demonstrates how to install IBM HTTP Server (IHS), install the Web Server Plug-in for WebSphere Application Server, and apply the interim fix for Dynamic Routing. You must enable IHS first before

setting up the Dynamic Routing feature or using auto scaling with Liberty collectives. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content shown in the video.

## Enabling IHS for Liberty Dynamic Routing

*Table 21. Demo installing IBM Installation Manager.* Show installing IBM Installation Manager.

Scene	Audio	Onscreen Action
1	This video shows how to install and enable IHS for Dynamic Routing using the IBM Installation Manager.	Show title Enabling IHS for Liberty Dynamic Routing.
2	Download and install the latest version of the IBM Installation Manager. You can use the Installation Manager to access online product repositories to install the Web Server Plug-in for WebSphere Application Server and the needed interim fix for the Dynamic Routing feature.	Direct user to the IBM Installation Manager and begin installing the latest version.
3	Once I select the latest version to install I click <b>Download package</b> from the installation menu. I then selected the platform I wanted to download and the appropriate fix for my machine. In this demo I will be installing the IBM Installation Manager Kit for Windows 64. You will be prompted for your IBM user ID and password before you can proceed with the download.	Show the user how to install the appropriate IBM Installation Kit from the IBM support site.
4	Select your download options. In this demo I will download the IBM Installation Manager using my browser (HTTPS). Click <b>Continue</b> . Download the provided zip file to your machine. You can locate this zip file in your downloads folder. Right click on the zip file in your downloads folder and click <b>Extract Files...</b> You must then select the location where you want to install the contents of the zip file.	Show the user how to download the IBM Installation Manager using a browser. Show the user how to download the zip file to their machine and extract the content to a new designated location.
5	When all of the files have been extracted successfully, open the folder containing the files and click <b>install.exe</b> The IBM Installation Manager will open and begin downloading. Accept the terms in the license agreements, and click <b>Next</b> . Verify that the install package is downloading in the appropriate directory, and click <b>Next</b> . Click <b>Install</b> . The IBM Installation Manager will now be installed.	Show the user how to run the install from the <code>install.exe</code> file. The user must accept the terms and license agreements to proceed with the install. Once they click Install the IBM Installation Manager will begin installing.
6	You will see a confirmation window when the install has been completed successfully. On this Window click <b>Restart Installation Manager</b> . Once the restart is complete, you can begin using the IBM Installation Manager.	Show the user the confirmation window that confirms the IBM Installation Manager was successfully downloaded. Show the user how to restart the IBM Installation Manager prior to using.

Table 22. Demo Set preferences for repository locations. Show how to set preferences for repository locations.

Scene	Audio	Onscreen Action
7	Start the Installation Manager and click <b>File</b> . Then click <b>Preferences</b> . Then click <b>Add Repository</b> to add the repository URL to the product. You can access repository URLs for the product via the Passport Advantage Online. Follow the same steps to add the repository URL needed to install the Web Server Plug-in for WebSphere Application Server	<p>Show user how to start the Installation Manager and add repository URLs. Users can gain access to repository URLs via the Passport Advantage Online. For more information on how to download repository URLs from Passport Advantage, go to the IBM Support Portal.</p> <p><b>IBM Support Portal-</b>  <a href="http://www-01.ibm.com/support/docview.wss?uid=swg27038625">http://www-01.ibm.com/support/docview.wss?uid=swg27038625</a></p> <p>Show the user how to add the following repository URL for installing the Web Server Plug-in.</p> <p><b>Web Server Plug-in for WebSphere Application Server</b> <a href="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLGILAN.v85">http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLGILAN.v85</a></p>

Table 23. Demo installing IBM HTTP Server. Show installing IBM HTTP Server.

Scene	Audio	Onscreen Action
8	Start the Installation Manager and click <b>Install</b> . In the <b>Install Packages</b> window select the appropriate version of <b>IHS</b> . Click <b>Next</b> . Accept the terms in the license agreements, and click <b>Next</b> .	Show users how to start the Installation Manger and locate IHS in the <b>Install Packages</b> window. Show them how to begin the installation and accept terms in the license agreements.
9	On the install packages window, specify the installation directory where you would like to install IHS. Click <b>Next</b> . On the next window IHS will already be selected for you. Click <b>Next</b> .	<p>Show the user an example of how to specify the installation directory. Note the following restrictions:</p> <ul style="list-style-type: none"> <li>• Deleting the default target location and leaving an installation-directory field empty prevents you from continuing. Do not use symbolic links as the destination directory.</li> <li>• Symbolic links are not supported.</li> <li>• Be careful on using spaces in the name of the installation directory. Some operating systems allow spaces in the specification of the installation directory. Other operating systems do not allow spaces.</li> <li>• Do not use a semicolon in the directory name. IBM HTTP Server cannot install properly if the target directory includes a semicolon.</li> </ul>
10	On the Configuration for IHS panel, specify your web server configuration. Specify a port number on which IHS will communicate. The default port is 80. Choose whether to use a Windows service to run IHS. Click <b>Next</b> .	Show users how to specify the web server configuration and specify the port number for IHS.
11	Review the summary information, and click <b>Install</b> . You will see a confirmation window if the installation was successful. Click <b>Finish</b> . <b>Note:</b> You don't have to start IHS until you're ready to use the Dynamic Routing or Auto Scaling features.	Show the user an example of summary information and the final step of installation. If the installation is successful, a message appears indicating the installation completed successfully.

Table 24. Demo installing Web Server Plug-in for WebSphere Application Server. Show installing Web Server Plug-in for WebSphere Application Server using IBM Installation Manager.

Scene	Audio	Onscreen Action
12	You already added the repository URL needed to install the Web Server Plug-in. Start the Installation Manager and click <b>Install</b> . In the <b>Install Packages</b> window select the appropriate version of the <b>Web Server Plug-in for WebSphere Application Server</b> . Click <b>Next</b> . Accept the terms in the license agreements, and click <b>Next</b> .	Show users how to start the Installation Manger and locate IHS in the <b>Install Packages</b> window. Show them how to begin the installation and accept terms in the license agreements.
13	On the install packages window, specify the installation directory where you would like to install the Web Server Plug-in. Click <b>Next</b> . On the next window the Web Server Plug-in will already be selected for you. Click <b>Next</b> .	Show the user an example of how to specify the installation directory. Note the following restrictions: <ul style="list-style-type: none"> <li>• Deleting the default target location and leaving an installation-directory field empty prevents you from continuing. Do not use symbolic links as the destination directory.</li> <li>• Symbolic links are not supported.</li> <li>• Be careful on using spaces in the name of the installation directory. Some operating systems allow spaces in the specification of the installation directory. Other operating systems do not allow spaces.</li> <li>• Do not use a semicolon in the directory name. IBM HTTP Server cannot install properly if the target directory includes a semicolon.</li> </ul>
14	Review the summary information, and click <b>Install</b> . You will see a confirmation window if the installation was successful. Click <b>Finish</b> .	Show the user an example of summary information and the final step of installation. If the installation is successful, a message appears indicating the installation completed successfully.

Table 25. Demo how to update the IBM HTTP Server. Updating the IBM HTTP Server

Scene	Audio	Onscreen Action
15	Start the Installation Manager. Click <b>Update</b> . Select the package group to update. Click <b>Next</b> . Select the version you want to update under <b>IHS</b> .	Show users how to locate and apply the interim fix for Dynamic Routing using the Installation Manager and IHS.
16	Review the summary information, and click <b>Update</b> . You will see a confirmation window if the installation was successful. Click <b>Finish</b> .	Show the user an example of summary information and the final step of installation. Done successfully a message will appear indicating a successful installation.

Table 26. Conclusion. Show where to find more information about Dynamic Routing and auto scaling.

Scene	Audio	Onscreen Action
17	For information on configuring clustered Liberty collectives for use with Dynamic Routing or Auto Scaling, see WASdev and the WebSphere Application Server Liberty documentation on IBM Knowledge Center.	Show information on documentation: <p><b>WASdev</b>  <a href="https://developer.ibm.com/wasdev/">https://developer.ibm.com/wasdev/</a></p> <p><b>WebSphere Application Server Liberty documentation on IBM Knowledge Center</b>  <a href="http://www-01.ibm.com/support/knowledgecenter/">http://www-01.ibm.com/support/knowledgecenter/</a></p>

For more information, see .

# Video: Getting started with the Server Configuration Tool for WebSphere Liberty

8.5.5.8

The following transcript is for the “Getting started with the Server Configuration Tool for WebSphere Liberty” video, which demonstrates how to enable and use the Liberty Admin Center Server Config tool. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

## Getting started with the Server Configuration Tool for WebSphere Liberty

Table 27. Title page. Show title and describe the video contents.

Scene	Audio	Onscreen Action
1	This video demonstrates the new web-based WebSphere Liberty server configuration tool. This tool makes it possible to view and edit server configuration files from a web browser.	Show title “Getting started with the Server Configuration Tool for WebSphere Liberty”.

Table 28. Demo enabling the Server Config tool. Show how to enable the Server Config tool.

Scene	Audio	Onscreen Action
2	To enable it, simply add the adminCenter-1.0 feature, keyStore, and quickStartSecurity elements to your server.xml file.  Optionally, if you are planning to use the tool to make changes to the server configuration, add the remote file access element.	Show a server.xml file with elements needed to enable the Server Config tool and enable editing: <pre>&lt;server&gt;   &lt;featureManager&gt;     &lt;feature&gt;adminCenter-1.0&lt;/feature&gt;   &lt;/featureManager&gt;    &lt;keyStore password="samplePassword"/&gt;    &lt;quickStartSecurity userName="sampleUser" userPassword="samplePassword"/&gt;    &lt;remoteFileAccess&gt;     &lt;writeDir&gt;\${server.config.dir}&lt;/writeDir&gt;   &lt;/remoteFileAccess&gt; &lt;/server&gt;</pre>
3	After saving these changes, a URL will appear in the server console.  Open this URL with your favorite web browser.	Show a console message and selecting the URL to the tool: Starting server configuration update Web application available: http://localhost:9080/ibm/adminCenter/serverConfig-1.0/ The server is ready to run a smarter planet.
4	Log in with the quickStartSecurity credentials.	Show entering sampleUser and samplePassword on the login page of Admin Center.

Table 29. Demo using the Server Config tool. Show how to view and edit server configuration files with the Server Config tool.

Scene	Audio	Onscreen Action
5	The file selection screen shows all of the configuration files available on the server. If nested include files are found, these will be hierarchically shown in the form of a tree.	Show the Configuration Files page of the Server Config tool.  Show expanding the primary list of files under server.xml.

Table 29. Demo using the Server Config tool (continued). Show how to view and edit server configuration files with the Server Config tool.

Scene	Audio	Onscreen Action
6	To edit a file simply click on its name. This brings up the main editing screen.  The tree shows all of the configuration elements in the document. Clicking on a tree node displays all of its associated values.  Values can be edited by typing on the text fields or using the selection buttons. If a mistake is made, automatic validation provides instant feedback.	Show selecting <code>server.xml</code> to display the contents of the <code>server.xml</code> file in <b>Design</b> mode.  Show selecting the <b>HTTP Endpoint</b> element to display its values.  Show selecting a new value, the display of a warning message, and selecting <b>x</b> to delete the value change.
7	Adding new elements is very easy. Simply click on the <b>Add child</b> button and select the element to be added.	Show selecting <b>Add child</b> , an element, and <b>Add</b> .
8	Removing elements is just as easy. Simply click on the <b>Remove</b> button and provide confirmation.	Show selecting <b>Remove</b> and then <b>Remove</b> to confirm the removal.
9	The order of the elements can be changed at any time using drag-and-drop gestures.	Show dragging an element to a different location in the tree.
10	The source of the document can be accessed by clicking the <b>Source</b> tab.	Show selecting the <b>Source</b> tab to display the <code>server.xml</code> source.
11	Syntax highlight, hover information, and content assist features are available.	Show hovering a cursor over an element value in the source to display information about the value.
12	After changes are complete, click on the <b>Save</b> button to persist them on the server.	Show selecting <b>Save</b> .

Table 30. Conclusion. State that the video is now ending.

Scene	Audio	Onscreen Action
13	This concludes the WebSphere Liberty server configuration tool video. Thanks for watching!	Show Copyright 2015 IBM Corporation. Voice over provided by Colleen Anderson. and then Thanks for watching!

For more information, see “Editing server configuration files in Admin Center” on page 1081.

## Video: Google OpenID Connect for applications on WebSphere Liberty

8.5.5.5

The following transcript is for the “Google OpenID Connect for applications on WebSphere Liberty” video, which demonstrates how to set up an OpenID Connect web single-sign-on on WebSphere Application Server Liberty with a Google OpenID Connect provider. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.


 Google OpenID Connect for applications on WebSphere Liberty

Table 31. Title page. Show title and then a basic Google OpenID Connect scenario.

Scene	Audio	Onscreen Action
1	This video will show you how to set up OpenID Connect web single-sign-on on WebSphere Application Server Liberty with a Google OpenID connect provider.	Show title <i>OpenID Connect Quick Setup with Google</i> .
2	Here you can see an “OpenID Connect” flow from an end user to an application on the Liberty server and the Google OpenID provider. When a user first attempts to access a Google OpenID Connect-protected application on a Liberty server, the user is redirected to the Google OpenID Provider. By using the Google account, the user is authenticated to access the protected web application on the Liberty server. In this video, we call a Liberty server, the “Relying party” or RP, and call “Google OpenID Connect provider” the OP.	Show a basic Google OpenID Connect scenario, that includes a Relying Party (RP), Google OpenID Connect provider (OP), and an End-User.

Table 32. Demo registering Liberty in Google

Scene	Audio	Onscreen Action
3	<p>To set up the Liberty RP with Google OP, first, we will register the Liberty server as an OpenID Connect client in the Google OP.</p> <p>To do so, we will</p> <ul style="list-style-type: none"> <li>• Log in to the Google developers console and create a project</li> <li>• Then in the project, we will create a “Client ID” for the Liberty server</li> <li>• Write down the Client ID and Client Secret for when we set up Liberty</li> </ul> <p>Let's try these steps now.</p>	<p>Show title <i>Register Liberty in Google</i>.</p> <ol style="list-style-type: none"> <li>1. On Google developer's console, create a project <a href="https://console.developers.google.com">https://console.developers.google.com</a>.</li> <li>2. In the project, create a Client ID from the <b>Credential</b> menu.</li> <li>3. Write down the following information for the Liberty setup <ul style="list-style-type: none"> <li>• Client ID</li> <li>• Client Secret</li> </ul> </li> </ol> <p>For more information, please refer to this page <a href="https://developers.google.com/accounts/docs/OpenIDConnect">https://developers.google.com/accounts/docs/OpenIDConnect</a>.</p>
4	In the Google Developers Console, create a new Project.	<p>In the Google Developers Console, we show a demo creating a new project.</p> <ul style="list-style-type: none"> <li>• Project name- WebSphereLibertyOpenIDConnect</li> <li>• Project ID- astute-tome-859</li> </ul>
5	In the project that you just created, go to <b>APIs &amp; auth</b> , then <b>Credentials</b> , and <b>Create new Client ID</b> . First, you will have to configure a consent screen.	Show the Google Developers Console screen where the <b>Create new Client ID</b> is selected.



Table 32. Demo registering Liberty in Google (continued)

Scene	Audio	Onscreen Action
6	The consent screen is shown to users when they authenticate with the Google OpenID provider. Configure your consent screen as needed and continue creating your Client ID. For the application type, select <b>Web application</b> . Then, enter a redirect URI for the Liberty server. (pointing to <code>https://rp-example.rtp.raleigh.ibm.com:7778/oidcclient/redirect/oidcRP</code> on the screen) This redirect URI comes from the configuration for your Liberty server, which we will cover later. If you don't know the redirect URI for your server, you can leave the default value and update it later.	Show the Google Developers Console screen where the <b>Web Application</b> is selected. <ul style="list-style-type: none"> <li>Authorized JavaScript Origins is set to <code>https://www.example.com</code></li> <li>Authorized Redirect URIs is set to <code>https://www.example.com/oauth2callback</code></li> </ul>
7	After the Client ID is created, you can see the Client ID and Client Secret. Make note of these values, because they are needed in the next step, configuring the Liberty server.	Show the Google Developers Console screen where the Client ID and Client Secret values can be seen.

Table 33. Demo configuring WebSphere Liberty

Scene	Audio	Onscreen Action
8	To set up Liberty to work with a Google OP, you will need to: <ul style="list-style-type: none"> <li>Install Version 8.5.5.5 or later of Liberty</li> <li>Install the OpenID Client feature</li> <li>Create a Liberty server</li> <li>Edit the <code>server.xml</code> configuration file with Google information</li> <li>Install the application that will use the Google account for authentication</li> <li>And finally, import the Google certificate into the keystore for SSL communication</li> </ul>	Show the WebSphere Liberty setup overview. <ol style="list-style-type: none"> <li>Install WebSphere Liberty 8.5.5.5 or latest beta &gt; <b>java -jar wlp-developers-runtime-8.5.5.5.jar</b></li> <li>Install OpenID Client feature (No download necessary) &gt; <b>bin/featureManager install openidConnectClient-1.0 --when-file-exists=ignore</b></li> <li>Create a Liberty server &gt; <b>server create GoogleRP</b></li> <li>Edit the <code>server.xml</code> with more configurations (Sample downloadable) <ul style="list-style-type: none"> <li>Required features</li> <li>SSL keystore</li> <li>OpenID Client</li> <li>Application</li> </ul> </li> <li>Install application that will use Google account for authentication &gt; Copy application ear/war file under app directory.</li> <li>Import Google certificate into keystore for SSL communication.</li> </ol>
9	First, we will install Version 8.5.5.5 of Liberty.  Then, we will install the OpenID Client feature, and create a server with the name GoogleRP.  You can find the <code>server.xml</code> configuration file under the <code>wlp\usr\servers\GoogleRP\</code> directory.	Demo with command prompt that is being used to update <code>server.xml</code> file.
10	Here is the default <code>server.xml</code> file. Now, we will compare it to a <code>server.xml</code> file that has a Google configuration.	Show a default <code>server.xml</code> file.

Table 33. Demo configuring WebSphere Liberty (continued)

Scene	Audio	Onscreen Action
11	<p>You can see that the necessary features are added. In the OpenID Connect Client configuration, the Client ID and Client Secret that we obtained from Google are added. You can obtain the other values by going to Google OP's discovery endpoint. (<a href="https://accounts.google.com/.well-known/openid-configuration">https://accounts/google.com/.well-known/openid-configuration</a> is shown on the video). Then we add an SSL configuration and end-point configuration with the host name, HTTP port, and HTTPS port.</p> <p>The configuration file also includes configuration for applications that rely on Google to perform authentication.</p> <p>That's all the configuration we need for Liberty.</p> <p>The Liberty RP uses this pattern <code>https://&lt;hostname&gt;:&lt;sslport&gt;/oidcclient/redirect/&lt;openidConnectClient id&gt;</code> to generate its own redirect URL. For example, the server that we configured has the following URI, <code>https://rp-example.rtp.raleigh.ibm.com:7778/oidcclient/redirect/oidcRP</code>. This is the URI that we entered earlier in the Google console.</p>	<p>Show a <code>server.xml</code> file that contains the Client ID and Client Secret that were obtained from Google. Also an SSL configuration and end-point configuration with the host name, HTTP port, and HTTPS port. The <code>server.xml</code> file also includes configuration for applications that rely on Google to perform authentication.</p>
12	<p>Next, we will install our application in the app directory.</p> <p>We will start and stop the Liberty server to get the keystore in the server resources and make sure that the Liberty server keystore has a Google certificate for SSL communication.</p> <p><b>Note:</b> We are not going to show cert steps in this video and include instruction in reference page</p> <p>Then we will start the Liberty server again.</p>	<p>Show title <i>How to import Google certificate here.</i></p>

Table 34. Demo testing the setup

Scene	Audio	Onscreen Action
13	<p>Now we will test our configuration to see if it works.</p> <ul style="list-style-type: none"> <li>• In a browser, we will go to the application URL.</li> <li>• When prompted, we will enter our Google account information.</li> <li>• We will see the application redirecting to Google to perform authentication.</li> <li>• After the user is authenticated, the RP will show an application page to the user.</li> <li>• Let's try that now.</li> </ul>	<p>Demo testing the setup.</p> <ol style="list-style-type: none"> <li>1. Start the WebSphere Liberty server &gt; <b>server start oidcRP</b></li> <li>2. In a browser, point to the application login page on the Liberty Server &gt; <code>http://rp-example.rtp.raleigh.ibm.com:7777/testpage</code></li> <li>3. When prompted, enter the Google user ID and password &gt; <code>xxx.yyy@gmail.com / mypassword</code></li> <li>4. Application relies on Google to perform authentication</li> <li>5. User is successfully authenticated</li> </ol>

Table 34. Demo testing the setup (continued)

Scene	Audio	Onscreen Action
14	In the browser, we will type in the URL of the application that is running on the Liberty server. Notice that we are prompted by the Google OP server because the Liberty relying party is delegating the authentication to the OP. We will enter the credentials for the Google account. After accepting the consent screen, we are successfully logged in to the application on the RP using the OP account.	Demo with a browser login that shows a successful login into the application on the RP using the OP account.

Table 35. Conclusion. Show where to find more information about OpenID Connect that uses Google.

Scene	Audio	Onscreen Action
15	For more information, visit these online resources.	<p>Show information on documentation:</p> <p><b>WebSphere Liberty download page</b>  <a href="https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-environments/">https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-environments/</a></p> <p><b>OpenID Connect feature installation</b>            Server: <a href="https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectServer-1.0">https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectServer-1.0</a>            Client: <a href="https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectClient-1.0">https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectClient-1.0</a></p> <p><b>IBM Knowledge Center - OpenID Connect main page</b>  <a href="http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-us/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_using_oidc.html">http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-us/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_using_oidc.html</a></p> <p><b>All OpenID Connect attributes are discussed here</b>  <a href="http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/twlp_config_oidc_rp.html?cp=SSEQTP_8.5.5%2F1-3-11-0-4-2-9-2">http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/twlp_config_oidc_rp.html?cp=SSEQTP_8.5.5%2F1-3-11-0-4-2-9-2</a></p> <p><b>IBM DeveloperWorks OpenID Connect article</b>  <a href="http://www.ibm.com/developerworks/websphere/library/techarticles/1502_odonnell/1502_odonnell.html">http://www.ibm.com/developerworks/websphere/library/techarticles/1502_odonnell/1502_odonnell.html</a></p> <p><b>WebSphere Liberty OpenID Connect setup video on YouTube</b>  <a href="http://youtu.be/fuajCS5bG4c">http://youtu.be/fuajCS5bG4c</a></p> <p><b>Google setup "OpenID Connect (OAuth 2.0 for Login)"</b> <a href="https://developers.google.com/accounts/docs/OpenIDConnect">https://developers.google.com/accounts/docs/OpenIDConnect</a></p>

For more information about OpenID Connect, see [8.5.5.5](#) Using OpenID Connect.

## Video: Installing Liberty from a ZIP file

8.5.5.6

The following transcript is for the “Installing Liberty from a ZIP file” video, which describes how you can quickly install Liberty from a ZIP archive file, start the server and add applications, and upgrade to a supported installation. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

### Installing Liberty from a ZIP file

*Table 36. Installing and upgrading Liberty.* Show how you can quickly install IBM WebSphere Application Server Liberty from a ZIP archive file, start the server and add applications, and upgrade to a supported installation.

Scene	Audio	Onscreen Action
1	Installing WebSphere Application Server Liberty can be as easy as extracting a ZIP file!	Show words, “Installing Liberty from a ZIP file”
2	To start, download one of the Liberty runtime ZIP files from IBM Fix Central or WASdev.net. You can choose from several prepackaged archives according to your needs, from just the Liberty kernel, which has a download size of less than 15 megabytes, to the Java Platform, Enterprise Edition 7 full platform or Web Profile, which you can optionally download with IBM Java 8.	Show the ZIP files on IBM Fix Central and WASdev.net.
3	After you download a ZIP file, just extract it to a local directory to install Liberty.	Show extracting the wlp-webProfile7-8.5.5.6 ZIP archive file to the C:\wlp-webProfile7-8.5.5.6 directory.
4	This example shows the Java EE 7 Web Profile ZIP, which comes preconfigured with Liberty features that support the Java EE 7 Web Profile specification.	Show running the <b>productInfo featureInfo</b> command on the command line. A list of installed features is displayed.
5	You can install additional features and other content using the <b>installUtility</b> command. The find option displays a list of installable content.	Show running the <b>installUtility find</b> command on the command line. A list of installable features, addons, samples, and other content is displayed.
	Start the Liberty server with the <b>server start</b> command.	Show running the <b>server start</b> command on the command line. The default server, defaultServer, starts.

Table 36. Installing and upgrading Liberty (continued). Show how you can quickly install IBM WebSphere Application Server Liberty from a ZIP archive file, start the server and add applications, and upgrade to a supported installation.

Scene	Audio	Onscreen Action
	<p>You can add an application at any time by placing the application WAR file in the dropins directory - no restart required!</p> <p>The Liberty server automatically starts the application when you add it to the directory.</p> <p>Standard server output and errors are captured in the console.log file; for example, this application outputs the URL of the web application, which can then be pasted into the browser.</p>	<p>Show copying the planningpoker-1.0-SNAPSHOT.war sample application into the wlp\usr\servers\defaultServer\dropins directory.</p> <p>Show opening the console.log file in the wlp\usr\servers\defaultServer\logs directory to copy the URL of the web application and pasting the URL into the browser. The web application loads successfully.</p>
	<p>Installing from the ZIP files enables no-charge, unsupported, unlimited use of Liberty in development environments and unsupported limited use in small-scale test and production environments.</p>	<p>Show running the <b>productInfo version</b> command on the command line. The product name, version, and edition is displayed. Because the installation is an unsupported version, the edition is BASE_ILAN.</p>

Table 36. Installing and upgrading Liberty (continued). Show how you can quickly install IBM WebSphere Application Server Liberty from a ZIP archive file, start the server and add applications, and upgrade to a supported installation.

Scene	Audio	Onscreen Action
	<p>For access to guaranteed service levels and IBM support, you can later upgrade to a supported edition from your existing installation. After you purchase WebSphere Application Server, just download and run the self-extracting license JAR file from Passport Advantage®.</p> <p>Note that if you are already a WebSphere Application Server customer - for example, if you are running Network Deployment traditional - you do not need to purchase anything extra; the Liberty entitlement is included.</p>	<p>Show copying the <code>wlp-nd-license.jar</code> file into the <code>C:\wlp-webProfile7-8.5.5.6</code> directory.</p> <p>Show running the <code>java -jar wlp-nd-license.jar</code> command on the command line to run the self-extracting JAR file. The command updates the license.</p> <p>Show running the <b>productInfo version</b> command, which now displays the ND edition.</p>
	<p>For more information about installing and upgrading Liberty, see the Knowledge Center documentation or WASdev.net.</p>	<p>Show words, “For more information about installing and upgrading Liberty, see:”</p> <ul style="list-style-type: none"> <li>• IBM Knowledge Center</li> <li>• WASdev.net</li> </ul>

For more information, see “Installing Liberty by extracting a ZIP archive file” on page 842.

## Video: Java EE 7 in Liberty

8.5.5.6

The following transcript is for the “Java EE 7 in Liberty ” video, which describes Liberty support for the Java Platform, Enterprise Edition (Java EE) 7 specifications and highlights ways to enable your Liberty servers for Java EE 7. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

### Java EE 7 in Liberty

Table 37. Title page and benefits of Java EE 7. Show title and then list the benefits of Java EE 7.

Scene	Audio	Onscreen Action
1	<p>Liberty now complies with Java Platform, Enterprise Edition Version 7. This video tells you about Liberty support for Java EE 7 and shows how you can quickly configure your servers for it.</p>	<p>Show title <i>Java EE 7 in Liberty</i> and the Java Compatible Enterprise Edition logo. Also show <i>Java Platform, Enterprise Edition Version 7</i> instead of <i>Java EE 7</i> for a few seconds to give the full name for <i>Java EE 7</i>.</p>

Table 37. Title page and benefits of Java EE 7 (continued). Show title and then list the benefits of Java EE 7.

Scene	Audio	Onscreen Action
2	<p>With Java EE 7, you have an open framework that enables you to provide robust business solutions and leverage your Java programming.</p> <p>You can deliver HTML5 dynamic scalable applications for desktops, tablets, and smartphones.</p> <p>You can be more productive. The simplified application architecture reduces the amount of boilerplate code needed for business logic.</p> <p>And you can support more enterprise demands. You can write batch applications in Java that use a standard API and are portable across multiple runtimes. You also can break down batch jobs into manageable chunks for uninterrupted performance.</p>	<p>Show animation that describes Java EE 7 and illustrates its main benefits:</p> <ul style="list-style-type: none"> <li>• HTML5 dynamic scalable applications</li> <li>• Increased developer productivity</li> <li>• Enterprise technologies such as batch processing</li> </ul>

Table 38. Grouping of specifications into "Java EE 7 full platform" and "Java EE 7 Web Profile". Show what specifications are available in the "Java EE 7 full platform" and "Java EE 7 Web Profile" groups.

Scene	Audio	Onscreen Action
3	<p>Java EE 7 introduces the full platform. All Java EE 7 specifications (or JSRs) are in the full platform.</p>	<p>Show image that has the entire Java EE 7. Highlight "Full Platform" and then all specifications.</p>

Table 38. Grouping of specifications into "Java EE 7 full platform" and "Java EE 7 Web Profile" (continued). Show what specifications are available in the "Java EE 7 full platform" and "Java EE 7 Web Profile" groups.

Scene	Audio	Onscreen Action																																																															
4	<p>Specifications for web applications are in the Web Profile, a subset of the full platform.</p> <p>Java EE 6 introduced Web Profile to assist developers of dynamic web applications, providing technologies such as EJB Lite, Java Persistence API, and Java Transaction API.</p> <p>For Java EE 7, Web Profile adds support for HTML5.</p> <p>Two new technologies, WebSocket and JSON, speed up data exchanges and simplify data parsing for portable applications. Updates to existing technologies, JAX-RS 2.0, Java Server Faces 2.2, and Servlet 3.1 enhance your ability to develop dynamic HTML5 applications.</p> <p>A more robust POJO development model enables broader use of annotations, such as in Interceptors and CDI. Bean Validation 1.1 offers method-level validation.</p>	<p>Show image that has the entire Java EE 7. Then show animation that lists the specifications in Web Profile and highlights the specifications named in the audio.</p> <p>Web Profile</p> <table border="1"> <thead> <tr> <th>Specification</th> <th>Java EE 6</th> <th>Java EE 7</th> </tr> </thead> <tbody> <tr><td>Bean Validation</td><td>1.0</td><td>1.1</td></tr> <tr><td>Common Annotations for the Java Platform</td><td>1.1</td><td>1.2</td></tr> <tr><td>Contexts and Dependency Injection (CDI)</td><td>1.0</td><td>1.2</td></tr> <tr><td>Debugging Support for Other Languages</td><td>1.0</td><td>1.0</td></tr> <tr><td>Dependency Injection for Java</td><td>1.0</td><td>1.0</td></tr> <tr><td>Enterprise JavaBeans (EJB) Lite</td><td>3.1</td><td>3.2</td></tr> <tr><td>Expression Language (JSP/EL)</td><td>2.2</td><td>3.0</td></tr> <tr><td>Interceptors</td><td>1.1</td><td>1.2</td></tr> <tr><td>Java API for JSON Processing (JSON-P)</td><td>n/a</td><td>1.0</td></tr> <tr><td>Java API for RESTful Web Services (JAX-RS)</td><td>n/a</td><td>2.0</td></tr> <tr><td>Java Database Connectivity (JDBC)</td><td>4.0</td><td>4.1</td></tr> <tr><td>Java Naming and Directory Interface (JNDI)</td><td>1.0</td><td>1.0</td></tr> <tr><td>Java Servlet</td><td>3.0</td><td>3.1</td></tr> <tr><td>JavaServer Faces (JSF)</td><td>2.0</td><td>2.2</td></tr> <tr><td>JavaServer Pages (JSP)</td><td>2.2</td><td>2.3</td></tr> <tr><td>Java Transaction API (JTA)</td><td>1.1</td><td>1.2</td></tr> <tr><td>Java Persistence API (JPA)</td><td>2.0</td><td>2.1</td></tr> <tr><td>Managed Beans</td><td>1.0</td><td>1.0</td></tr> <tr><td>Standard Tag Library for JavaServer Pages (JSTL)</td><td>1.2</td><td>1.2</td></tr> <tr><td>WebSocket</td><td>n/a</td><td>1.0, 1.1</td></tr> </tbody> </table>	Specification	Java EE 6	Java EE 7	Bean Validation	1.0	1.1	Common Annotations for the Java Platform	1.1	1.2	Contexts and Dependency Injection (CDI)	1.0	1.2	Debugging Support for Other Languages	1.0	1.0	Dependency Injection for Java	1.0	1.0	Enterprise JavaBeans (EJB) Lite	3.1	3.2	Expression Language (JSP/EL)	2.2	3.0	Interceptors	1.1	1.2	Java API for JSON Processing (JSON-P)	n/a	1.0	Java API for RESTful Web Services (JAX-RS)	n/a	2.0	Java Database Connectivity (JDBC)	4.0	4.1	Java Naming and Directory Interface (JNDI)	1.0	1.0	Java Servlet	3.0	3.1	JavaServer Faces (JSF)	2.0	2.2	JavaServer Pages (JSP)	2.2	2.3	Java Transaction API (JTA)	1.1	1.2	Java Persistence API (JPA)	2.0	2.1	Managed Beans	1.0	1.0	Standard Tag Library for JavaServer Pages (JSTL)	1.2	1.2	WebSocket	n/a	1.0, 1.1
Specification	Java EE 6	Java EE 7																																																															
Bean Validation	1.0	1.1																																																															
Common Annotations for the Java Platform	1.1	1.2																																																															
Contexts and Dependency Injection (CDI)	1.0	1.2																																																															
Debugging Support for Other Languages	1.0	1.0																																																															
Dependency Injection for Java	1.0	1.0																																																															
Enterprise JavaBeans (EJB) Lite	3.1	3.2																																																															
Expression Language (JSP/EL)	2.2	3.0																																																															
Interceptors	1.1	1.2																																																															
Java API for JSON Processing (JSON-P)	n/a	1.0																																																															
Java API for RESTful Web Services (JAX-RS)	n/a	2.0																																																															
Java Database Connectivity (JDBC)	4.0	4.1																																																															
Java Naming and Directory Interface (JNDI)	1.0	1.0																																																															
Java Servlet	3.0	3.1																																																															
JavaServer Faces (JSF)	2.0	2.2																																																															
JavaServer Pages (JSP)	2.2	2.3																																																															
Java Transaction API (JTA)	1.1	1.2																																																															
Java Persistence API (JPA)	2.0	2.1																																																															
Managed Beans	1.0	1.0																																																															
Standard Tag Library for JavaServer Pages (JSTL)	1.2	1.2																																																															
WebSocket	n/a	1.0, 1.1																																																															



Table 38. Grouping of specifications into "Java EE 7 full platform" and "Java EE 7 Web Profile" (continued). Show what specifications are available in the "Java EE 7 full platform" and "Java EE 7 Web Profile" groups.

Scene	Audio	Onscreen Action																																																																																																																											
5	<p>Also added for Version 7 are specifications for enterprise, web service, batch and other applications, as well as support for application security, deployment and management. These specifications are in the full platform.</p> <p>Java EE 7 has a simplified architecture requiring less boilerplate code for business logic, such as in JMS 2.0.</p> <p>For the enterprise, you can use Batch Applications to better utilize computing resources by shifting processing times to when resources are typically idle. Concurrency Utilities supports scalable applications that integrate with the Java EE runtime in a secure, reliable manner.</p> <p>The full platform also has updated support for Java Connector Architecture and Java Message Service.</p>	<p>Show image that has the entire Java EE 7. Then show animation that lists the specifications in the full platform and highlights the specifications named in the audio.</p> <p>Full Platform</p> <p>Web Profile</p> <table border="1"> <thead> <tr> <th>Specification</th> <th>Java EE 6</th> <th>Java EE 7</th> </tr> </thead> <tbody> <tr><td>Bean Validation</td><td>1.0</td><td>1.1</td></tr> <tr><td>Common Annotations for the Java Platform</td><td>1.1</td><td>1.2</td></tr> <tr><td>Contexts and Dependency Injection (CDI)</td><td>1.0</td><td>1.2</td></tr> <tr><td>Debugging Support for Other Languages</td><td>1.0</td><td>1.0</td></tr> <tr><td>Dependency Injection for Java</td><td>1.0</td><td>1.0</td></tr> <tr><td>Enterprise JavaBeans (EJB) Lite</td><td>3.1</td><td>3.2</td></tr> <tr><td>Expression Language (JSP/EL)</td><td>2.2</td><td>3.0</td></tr> <tr><td>Interceptors</td><td>1.1</td><td>1.2</td></tr> <tr><td>Java API for JSON Processing (JSON-P)</td><td>n/a</td><td>1.0</td></tr> <tr><td>Java API for RESTful Web Services (JAX-RS)</td><td>n/a</td><td>2.0</td></tr> <tr><td>Java Database Connectivity (JDBC)</td><td>4.0</td><td>4.1</td></tr> <tr><td>Java Naming and Directory Interface (JNDI)</td><td>1.0</td><td>1.0</td></tr> <tr><td>Java Servlet</td><td>3.0</td><td>3.1</td></tr> <tr><td>JavaServer Faces (JSF)</td><td>2.0</td><td>2.2</td></tr> <tr><td>JavaServer Pages (JSP)</td><td>2.2</td><td>2.3</td></tr> <tr><td>Java Transaction API (JTA)</td><td>1.1</td><td>1.2</td></tr> <tr><td>Java Persistence API (JPA)</td><td>2.0</td><td>2.1</td></tr> <tr><td>Managed Beans</td><td>1.0</td><td>1.0</td></tr> <tr><td>Standard Tag Library for JavaServer Pages (JSTL)</td><td>1.2</td><td>1.2</td></tr> <tr><td>WebSocket</td><td>n/a</td><td>1.0, 1.1</td></tr> </tbody> </table> <p>Remaining Full Platform</p> <table border="1"> <tbody> <tr><td>Batch Applications for Java Platform</td><td>n/a</td><td>1.0</td></tr> <tr><td>EE Concurrency Utilities</td><td>n/a</td><td>1.0</td></tr> <tr><td>Enterprise JavaBeans (EJB) full</td><td>n/a</td><td>3.2</td></tr> <tr><td>Implementing Enterprise Web Services</td><td>n/a</td><td>1.4</td></tr> <tr><td>J2EE Management</td><td>n/a</td><td>1.1</td></tr> <tr><td>Java API for RESTful Web Services (JAX-RS)</td><td>1.1</td><td>n/a</td></tr> <tr><td>Java API for XML-Based Web Services (JAX-WS)</td><td>n/a</td><td>2.2</td></tr> <tr><td>Java API for WSDL (JWSDL)</td><td></td><td></td></tr> <tr><td>Java API for XML Processing (JAXP)</td><td>n/a</td><td>1.4</td></tr> <tr><td>Java Architecture for XML Binding (JAXB)</td><td>n/a</td><td>2.2</td></tr> <tr><td>Java Authentication Service Provider Interface for Containers (JASPIC)</td><td>n/a</td><td>1.1</td></tr> <tr><td>Java Authorization Contract for Containers (JACC)</td><td>n/a</td><td>1.5</td></tr> <tr><td>Java EE Connector Architecture (JCA)</td><td>1.6</td><td>1.7</td></tr> <tr><td>JavaMail</td><td>n/a</td><td>1.5</td></tr> <tr><td>Java Message Service (JMS) API</td><td>1.1</td><td>2.0</td></tr> <tr><td>Java Management Extensions (JMX)</td><td>n/a</td><td>2.0</td></tr> <tr><td>JavaBeans Activation Framework (JAF)</td><td>n/a</td><td>1.1</td></tr> <tr><td>SOAP with Attachments API for Java (SAAJ)</td><td>n/a</td><td>1.3</td></tr> <tr><td>Streaming API for XML (StAX)</td><td>n/a</td><td>1.0</td></tr> <tr><td>Web Services Metadata for the Java Platform</td><td></td><td></td></tr> </tbody> </table>	Specification	Java EE 6	Java EE 7	Bean Validation	1.0	1.1	Common Annotations for the Java Platform	1.1	1.2	Contexts and Dependency Injection (CDI)	1.0	1.2	Debugging Support for Other Languages	1.0	1.0	Dependency Injection for Java	1.0	1.0	Enterprise JavaBeans (EJB) Lite	3.1	3.2	Expression Language (JSP/EL)	2.2	3.0	Interceptors	1.1	1.2	Java API for JSON Processing (JSON-P)	n/a	1.0	Java API for RESTful Web Services (JAX-RS)	n/a	2.0	Java Database Connectivity (JDBC)	4.0	4.1	Java Naming and Directory Interface (JNDI)	1.0	1.0	Java Servlet	3.0	3.1	JavaServer Faces (JSF)	2.0	2.2	JavaServer Pages (JSP)	2.2	2.3	Java Transaction API (JTA)	1.1	1.2	Java Persistence API (JPA)	2.0	2.1	Managed Beans	1.0	1.0	Standard Tag Library for JavaServer Pages (JSTL)	1.2	1.2	WebSocket	n/a	1.0, 1.1	Batch Applications for Java Platform	n/a	1.0	EE Concurrency Utilities	n/a	1.0	Enterprise JavaBeans (EJB) full	n/a	3.2	Implementing Enterprise Web Services	n/a	1.4	J2EE Management	n/a	1.1	Java API for RESTful Web Services (JAX-RS)	1.1	n/a	Java API for XML-Based Web Services (JAX-WS)	n/a	2.2	Java API for WSDL (JWSDL)			Java API for XML Processing (JAXP)	n/a	1.4	Java Architecture for XML Binding (JAXB)	n/a	2.2	Java Authentication Service Provider Interface for Containers (JASPIC)	n/a	1.1	Java Authorization Contract for Containers (JACC)	n/a	1.5	Java EE Connector Architecture (JCA)	1.6	1.7	JavaMail	n/a	1.5	Java Message Service (JMS) API	1.1	2.0	Java Management Extensions (JMX)	n/a	2.0	JavaBeans Activation Framework (JAF)	n/a	1.1	SOAP with Attachments API for Java (SAAJ)	n/a	1.3	Streaming API for XML (StAX)	n/a	1.0	Web Services Metadata for the Java Platform		
Specification	Java EE 6	Java EE 7																																																																																																																											
Bean Validation	1.0	1.1																																																																																																																											
Common Annotations for the Java Platform	1.1	1.2																																																																																																																											
Contexts and Dependency Injection (CDI)	1.0	1.2																																																																																																																											
Debugging Support for Other Languages	1.0	1.0																																																																																																																											
Dependency Injection for Java	1.0	1.0																																																																																																																											
Enterprise JavaBeans (EJB) Lite	3.1	3.2																																																																																																																											
Expression Language (JSP/EL)	2.2	3.0																																																																																																																											
Interceptors	1.1	1.2																																																																																																																											
Java API for JSON Processing (JSON-P)	n/a	1.0																																																																																																																											
Java API for RESTful Web Services (JAX-RS)	n/a	2.0																																																																																																																											
Java Database Connectivity (JDBC)	4.0	4.1																																																																																																																											
Java Naming and Directory Interface (JNDI)	1.0	1.0																																																																																																																											
Java Servlet	3.0	3.1																																																																																																																											
JavaServer Faces (JSF)	2.0	2.2																																																																																																																											
JavaServer Pages (JSP)	2.2	2.3																																																																																																																											
Java Transaction API (JTA)	1.1	1.2																																																																																																																											
Java Persistence API (JPA)	2.0	2.1																																																																																																																											
Managed Beans	1.0	1.0																																																																																																																											
Standard Tag Library for JavaServer Pages (JSTL)	1.2	1.2																																																																																																																											
WebSocket	n/a	1.0, 1.1																																																																																																																											
Batch Applications for Java Platform	n/a	1.0																																																																																																																											
EE Concurrency Utilities	n/a	1.0																																																																																																																											
Enterprise JavaBeans (EJB) full	n/a	3.2																																																																																																																											
Implementing Enterprise Web Services	n/a	1.4																																																																																																																											
J2EE Management	n/a	1.1																																																																																																																											
Java API for RESTful Web Services (JAX-RS)	1.1	n/a																																																																																																																											
Java API for XML-Based Web Services (JAX-WS)	n/a	2.2																																																																																																																											
Java API for WSDL (JWSDL)																																																																																																																													
Java API for XML Processing (JAXP)	n/a	1.4																																																																																																																											
Java Architecture for XML Binding (JAXB)	n/a	2.2																																																																																																																											
Java Authentication Service Provider Interface for Containers (JASPIC)	n/a	1.1																																																																																																																											
Java Authorization Contract for Containers (JACC)	n/a	1.5																																																																																																																											
Java EE Connector Architecture (JCA)	1.6	1.7																																																																																																																											
JavaMail	n/a	1.5																																																																																																																											
Java Message Service (JMS) API	1.1	2.0																																																																																																																											
Java Management Extensions (JMX)	n/a	2.0																																																																																																																											
JavaBeans Activation Framework (JAF)	n/a	1.1																																																																																																																											
SOAP with Attachments API for Java (SAAJ)	n/a	1.3																																																																																																																											
Streaming API for XML (StAX)	n/a	1.0																																																																																																																											
Web Services Metadata for the Java Platform																																																																																																																													

Table 38. Grouping of specifications into "Java EE 7 full platform" and "Java EE 7 Web Profile" (continued). Show what specifications are available in the "Java EE 7 full platform" and "Java EE 7 Web Profile" groups.

Scene	Audio	Onscreen Action
6	<p>In all, Java EE 7 has over 20 new or changed specifications.</p> <p>The Liberty product supports the full platform specifications, while the Liberty Core product supports mainly the Web Profile specifications.</p>	Show image of the entire Java EE 7. Highlight the specifications supported by Liberty and then Liberty Core.

Table 39. Demo installing Liberty with Java EE 7 by extracting a compressed (ZIP) file. Show how to install a Liberty runtime with Java EE 7 by extracting a ZIP file downloaded from WASdev.

Scene	Audio	Onscreen Action
7	You can install Liberty with Java EE 7 technologies by downloading a compressed, or ZIP, file from the WASdev website to a temporary directory, and then extracting the ZIP file to an empty directory.	Show how to download a ZIP file with Liberty and Java EE technologies from the WASdev website to C:\wlp_temp on a workstation and then extract the ZIP file to C:\, resulting in installation of Liberty to C:\wlp.
8	It's that simple!	Show selecting the C:\wlp installation directory.

Table 40. Demo adding a Liberty runtime with Java EE 7 features in WebSphere Developer Tools for Eclipse. Show how to install a server with Java EE features in WebSphere Developer Tools.

Scene	Audio	Onscreen Action
9	<p>In WebSphere Developer Tools for Eclipse, you can add a Liberty runtime with Java EE technologies.</p> <p>Create a new server and select to download and install a Liberty runtime environment from <a href="http://ibm.com">ibm.com</a>.</p> <p>The runtime options with Java EE 7 technologies are for the full platform, Web Profile or client.</p> <p>You can add on individual features. Technologies that are in the selected runtime option are greyed out.</p>	<p>Show images that demo how to add a Liberty server that has Java EE technologies in WebSphere Developer Tools.</p> <ol style="list-style-type: none"> <li>1. Right-click in the Servers view and select <b>New &gt; Server</b>.</li> <li>2. In the New Server wizard: <ol style="list-style-type: none"> <li>a. Select the <b>WebSphere Application Server Liberty</b> server type and click the <b>Add</b> link.</li> <li>b. Select <b>Install from an archive or repository</b> and click <b>Next</b>.</li> <li>c. Specify the location to which to install Liberty, select <b>Download and install a new runtime environment from <a href="http://ibm.com">ibm.com</a></b>, select a Liberty product with Java EE 7 technologies, and click <b>Next</b>.</li> <li>d. Select any add-ons to install and click <b>Next</b>.</li> <li>e. Accept the license agreement and click <b>Finish</b>.</li> <li>f. After installation, click <b>Next</b>.</li> <li>g. Specify a server name and click <b>Next</b>.</li> <li>h. Click <b>Finish</b>.</li> </ol> </li> </ol> <p>To start the server, right-click the Liberty server in the Servers view and click <b>Start</b>.</p>

Table 41. Demo adding Java EE 7 features to a Liberty installation from a command line and Installation Manager. Show how to run an `installUtility` command to install Java EE features. Briefly show the Installation Manager option.

Scene	Audio	Onscreen Action
10	If you already have Liberty installed, you can add Java EE 7 features to your installation by running a <code>featureManager</code> or <code>installUtility</code> command.	<p>Show running an <code>installUtility</code> command to install features into an existing installation of Liberty at C:\wlp.</p> <ol style="list-style-type: none"> <li>From a command line at C:\wlp\bin, show running a command to install the <code>webProfile-7.0</code> feature:  <pre>installUtility install webProfile-7.0</pre> </li> <li>Enter 1 to agree to the terms of the license agreement.</li> </ol> <p>Command messages list the features installed.</p>
11	You also can use Installation Manager to install Java EE 7 features.	<p>Show images that demo how to use Installation Manager to install Liberty with Java EE 7 features.</p> <ol style="list-style-type: none"> <li>During installation of IBM WebSphere Application Server Liberty Network Deployment 8.5.5.6, under <b>Liberty Repositories</b> on the Install Packages page, select <b>Allow Installation Manager to connect to the IBM WebSphere Liberty Repository</b> and click <b>Next</b>.</li> <li>Under <b>Asset Selection</b> on the Install Packages page, click <b>Launch Asset Selection Wizard</b>.</li> <li>In the Asset Selection dialog: <ol style="list-style-type: none"> <li>Click the <b>Install</b> button to select a Java EE technology to install.</li> <li>After the <b>Install</b> button changes to the <b>Install Pending</b> button, click <b>Next</b>.</li> <li>Under <b>License Agreement</b>, select <b>I accept the terms in the license agreement</b> and click <b>Finish</b>.</li> </ol> </li> <li>Under <b>Asset Selection</b> on the Install Packages page, review the list of assets to install and click <b>Next</b>.</li> </ol>
12		<p>Show a summary of the ways to install Java EE 7 technologies for Liberty:</p> <ul style="list-style-type: none"> <li>ZIP file from the WASdev website</li> <li>WebSphere Application Server Developer Tools for Eclipse</li> <li><code>installUtility</code> or <code>featureManager</code> command</li> <li>IBM Installation Manager</li> </ul>

Table 42. Demo configuration of a Liberty server to add a Java EE 7 feature. Show how to add a Java EE 7 feature to a server configuration.

Scene	Audio	Onscreen Action
13	After Java EE 7 features are installed, adding support for a Java EE 7 specification to a Liberty server is as simple as adding a feature name to a <code>server.xml</code> file.	<p>Under the heading <i>Configuration</i>, show a command line at C:\wlp\bin with the command <code>server run server1</code> and with messages indicating that <code>server1</code> is running. Also show a text editor open on the <code>server.xml</code> file of <code>server1</code>. Finally, show adding the <code>jaxrs-2.0</code> feature to a feature manager and the resulting <code>server1</code> messages that confirm the server configuration change.</p>
14	Liberty provides the <code>javaee-7.0</code> , <code>webProfile-7.0</code> , and <code>javaeeClient-7.0</code> convenience features to make it easier to enable your servers to support a broad range of applications.	<p>Show a list of the Liberty convenience features for Java EE 7:</p> <ul style="list-style-type: none"> <li><code>javaee-7.0</code></li> <li><code>webProfile-7.0</code></li> <li><code>javaeeClient-7.0</code></li> </ul>

Table 42. Demo configuration of a Liberty server to add a Java EE 7 feature (continued). Show how to add a Java EE 7 feature to a server configuration.

Scene	Audio	Onscreen Action
15	Use the <code>javaee-7.0</code> feature to quickly add support for all specifications. The <code>webProfile-7.0</code> feature adds support for web applications. And the <code>javaeeClient-7.0</code> feature allows you to quickly configure an application client component.	Show sample configuration files for the <code>javaee-7.0</code> , <code>webProfile-7.0</code> , and <code>javaeeClient-7.0</code> convenience features.

Table 43. Some features require configuration or migration. Show where to find instructions about the needed configuration or migration.

Scene	Audio	Onscreen Action
16	<p>The IBM Knowledge Center has information about the features.</p> <p><i>Java EE 7 programming model support</i> lists the Java EE specifications, provides links to the JSRs and Liberty features, and tells you what products support the specifications. Note that not all Java EE specifications have their own Liberty feature.</p> <p>For some of the features, you'll need to do configuration beyond adding the feature name to a <code>server.xml</code>.</p> <p>If your server uses Java EE 6 features and you are considering adding Version 7 features, look at <i>Supported Java EE 6 and 7 feature combinations</i>. Also, look at <i>Java EE 7 behavior changes</i> to see whether moving from a Version 6 feature to a Version 7 feature would benefit your applications and environment.</p> <p>For details on features, see <i>Liberty features</i>.</p>	<p>Show topics in Knowledge Center that identify and cover feature configuration and migration:</p> <ul style="list-style-type: none"> <li>• <i>Java EE 7 programming model support</i></li> <li>• <i>Supported Java EE 6 and 7 feature combinations</i></li> <li>• <i>Java EE 7 behavior changes</i></li> <li>• <i>Liberty features</i></li> </ul>

Table 44. Conclusion. Show where to find more information about Java EE 7 in Liberty.

Scene	Audio	Onscreen Action
17	For how-to articles and videos on using Java EE 7 in your applications as well as information about configuring servers, see <code>WASdev.net</code> and the WebSphere Application Server Liberty documentation on IBM Knowledge Center.	<p>Show where to find information about Java EE 7 in Liberty:</p> <p><b>WASdev</b>  <a href="http://developer.ibm.com/wasdev">http://developer.ibm.com/wasdev</a></p> <p><b>IBM Knowledge Center</b>  <a href="http://www.ibm.com/support/knowledgecenter/">http://www.ibm.com/support/knowledgecenter/</a></p>

## Video: OpenID Connect on Liberty

8.5.5.5

The following transcript is for the “OpenID Connect on Liberty” video, which demonstrates how to configure OpenID Connect on Liberty. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

 OpenID Connect on Liberty

Table 45. Title page. Show title and then a basic OpenID Connect scenario, along with supported OpenID providers and benefits of using OpenID Connect.

Scene	Audio	Onscreen Action
1	This video will show you how to set up a simple OpenID Connect web single-sign-on scenario using WebSphere Application Server Liberty.	Show title <i>OpenID Connect Quick Setup</i> .
2	<p>Here you can see a basic "OpenID Connect" flow. When a user first attempts to access an OpenID Connect-protected web application, or relying party, the user is redirected to an OpenID Connect provider. The OpenID Connect provider authenticates the user and obtains the user's authorization, then responds with an authorization code. The application container then extracts the code from the response, sends the code back to the OpenID provider for verification, and receives ID and access tokens. As a result, the user is authenticated to access the protected web application. Using the access token, the application can request user information, such as an email address, from the OpenID Connect provider, or it can access any service that supports OpenID Connect. In this video, I will refer to the application as the "Relying Party" or RP and the "OpenID Provider" as OP.</p> <p>Let's take a look at the several supported OpenID providers.</p>	Show a basic OpenID Connect scenario, that includes a Relying Party (RP), OpenID Provider (OP), and an End-User.
3	You can configure IBM WebSphere Liberty either as an OpenID provider or a relying party. You can use IBM Security Access Manager, also known as ISAM, as an OP as well. Alternatively, you can use a number of supported third-party OpenID providers.	<p>Show some of the supported OpenID providers.</p> <ul style="list-style-type: none"> <li>• IBM WebSphere</li> <li>• IBM Security Access Manager</li> <li>• Amazon</li> <li>• Microsoft</li> <li>• Okta</li> <li>• Google</li> </ul>

Table 45. Title page (continued). Show title and then a basic OpenID Connect scenario, along with supported OpenID providers and benefits of using OpenID Connect.

Scene	Audio	Onscreen Action
4	<p>OpenID Connect offers a number of benefits as an identity layer on top of OAuth 2.0. With OpenID Connect, users have a single internet identity that they can use to authenticate across several servers, services, and applications, and it reduces the amount of maintenance work in applications because they no longer need their own user registry.</p> <p>For developers, it simplifies the task of authenticating users without taking on the responsibility of storing and managing passwords. OpenID Connect can also extend security services to cloud-based and mobile applications written in any language, such as JavaScript, Ruby, node.js, or Java, and it can function as a single security manager for provisioning hundreds of Liberty servers in a cloud environment. Because OpenID Connect combines the advantages of identity, authentication, and OAuth, OpenID Connect is a significant improvement over OAuth alone.</p>	<p>Show some of the benefits of using OpenID Connect.</p> <ul style="list-style-type: none"> <li>• OpenID Connect makes it easier for a user to use a single internet identity (user account), to authenticate across several servers, services, and applications.</li> <li>• Applications no longer need to maintain their own user registry.</li> <li>• OpenID Connect extends security services to cloud and mobile applications, accessible through languages such as JavaScript, Ruby, node.js, Java</li> <li>• Provisioning hundreds of Liberty servers in a cloud offers the ability to have a single security manager</li> <li>• OpenID Connect is a significant improvement over OAuth 2.0</li> </ul>

Table 46. Demo configuring the OpenID Provider

Scene	Audio	Onscreen Action
5	<p>WebSphere Application Server Liberty can be configured as the OpenID provider, the relying party, or both. If you want to use Liberty as both an OP and RP, you must configure them on different Liberty server instances. We will set up Liberty servers as OP and RP and take a look at a simple web single-sign-on scenario between the Liberty OP and RP.</p>	<p>Show title <i>Setting up Liberty as OpenID Connect provider and relying party.</i></p>
6	<p>First, we will set up an OpenID provider.</p> <p>To do so, we will install WebSphere Liberty 8554 or later, which is required to use the OpenID Connect features. Install the OpenID Server feature</p> <p>Create a Liberty server and add an OP configuration to the server.xml file, which is available as a downloadable sample from IBM developerWorks.</p>	<p>Show OP setup overview.</p> <ol style="list-style-type: none"> <li>1. Install WebSphere Liberty 8.5.5.4 or above &gt; <b>java -jar wlp-developers-runtime-8.5.5.4.jar</b></li> <li>2. Install OpenID Server feature (No download necessary) &gt; <b>bin/featureManager install openidConnectServer-1.0 --when-file-exists=ignore</b></li> <li>3. Create a Liberty server &gt; <b>server create oidcServer</b></li> </ol> <ul style="list-style-type: none"> <li>• Edit server.xml with more configurations (Sample downloadable)</li> <li>• Required features</li> <li>• SSL keystore</li> <li>• User Registry</li> <li>• OpenID Server</li> </ul>

Table 46. Demo configuring the OpenID Provider (continued)

Scene	Audio	Onscreen Action
7	<p>First, unpack the Liberty JAR file. This creates the wlp directory. Go to the bin directory under wlp, and run the <b>featureManager install</b> command to install the OpenID Connect Server feature.</p> <p>In the same directory, run the <b>server create</b> command to create a Liberty OP server. We will name this one oidcServer.</p> <p>oidcServer is now created with the minimum configuration in the server.xml file. You can find the configuration in the wlp/usr/server/oidcServer directory.</p> <p>Here you can see the contents of the server.xml file that we just created. The configuration is very simple, just one feature and port information. We will replace it with a server.xml file that has the OP server configuration. (Screen splits and right-hand side OP config appears).</p> <p>(Going through updates in the server.xml file) In this OP server configuration,</p> <ul style="list-style-type: none"> <li>• Required features are added.</li> <li>• A host name is added.</li> <li>• Keystore configuration is included for the SSL feature.</li> <li>• The OP maintains user accounts, so a user registry is configured.</li> </ul> <p>The rest is OP configuration that uses OAuth technology. It includes information about the relying party that it performs authorization for.</p> <p>You can download the server.xml file that we just added from IBM DeveloperWorks. We will start the OP server. Now that we have set up the OpenID provider, we can set up the Liberty relying party.</p>	<p>Demo with a command prompt that is being used to update server.xml file.</p>

Table 47. Demo configuring the Relying Party

Scene	Audio	Onscreen Action
8	<p>To set up the relying party, just like the OP configuration, we need to have version 8.5.5.4 or later of the Liberty profile and we will install the OpenID Client feature.</p> <p>We will create a separate Liberty server and edit the server.xml file.</p> <p>Then we will install the application and exchange keys with the OpenID provider for SSL communication.</p>	<p>Show RP setup overview.</p> <ol style="list-style-type: none"> <li>1. Install WebSphere Liberty 8.5.5.4 &gt; <b>java -jar wlp-developers-runtime-8.5.5.4.jar</b></li> <li>2. Install OpenID Client feature (No download necessary) &gt; <b>bin/featureManager install openidConnectClient-1.0 --when-file-exists=ignore</b></li> <li>3. Create a Liberty server &gt; <b>server create oidcRP</b> <ul style="list-style-type: none"> <li>• Edit server.xml with more configurations (Sample downloadable)</li> <li>• Required features</li> <li>• SSL keystore</li> <li>• OpenID Client</li> <li>• Application</li> </ul> </li> </ol> <ol style="list-style-type: none"> <li>1. Install application (that uses OpenID Connect) &gt;&gt; Copy application ear/war file under app directory           <ul style="list-style-type: none"> <li>• Exchange keys with OP for SSL communication</li> </ul> </li> </ol>
9	<p>Version 8.5.5.4 of Liberty is already configured on this machine. We will install the OpenID Client feature and create a server with the name oidcRP. Here is the default server.xml file. Now, we will compare it to a server.xml file with an RP configuration. These sections - features, endpoint host name, keystores - are the same updates that we previously saw with the OP.</p> <p>This time, we have an OpenID Client configuration instead of the OP server configuration. It specifies the OP URLs to send authentication requests to.</p> <p>The RP configuration also includes application configuration and these applications rely on the OP to perform authentication.</p> <p>Note that there is no user registry configuration on the RP. That is all the configuration needed for the RP. We will copy a test application into the app directory of the RP. Before you start the RP server, make sure that the RP and OP exchanged the keys in the keystore for SSL communication. In this demo, we will use the same keystore and same password.</p> <p>Now we will start the RP server to see if we are set up correctly.</p>	<p>Demo with a command prompt that is being used to update server.xml file.</p>



Table 48. Demo testing the OP/RP setup

Scene	Audio	Onscreen Action
10	The OP and RP servers are already started. In a browser, we will go to the application URL. When prompted, we will enter account information from the OP. We will see the RP relying on the OP to perform authentication. Once the user is authenticated, the RP will show an application page to the user. Let's try that now.	<p>Demo testing the OP/RP setup.</p> <ol style="list-style-type: none"> <li>1. Start both OP and RP servers, &gt; <b>server start oidcServer</b> &gt; <b>server start oidcRP</b></li> <li>2. In a browser, point to the application login page on RP &gt; <a href="https://oidc-rp.rtp.raleigh.ibm.com:9443/testpage">https://oidc-rp.rtp.raleigh.ibm.com:9443/testpage</a></li> <li>3. When prompted, enter the user ID and password that is maintained by the OP &gt; user1 / security</li> <li>4. RP relies on OP for authentication</li> <li>5. Upon successful authentication, RP serves application page with user information.</li> </ol>
11	<p>In the browser, we will type in the application URL that is on the RP server. It will prompt for a user name and password. Notice that we are being prompted by the OP server because the RP is delegating the authentication to the OP.</p> <p>We will enter user1 and security, which are the credentials for the OP account. Now we are successfully logged in to the application on the RP using the OP account.</p>	<p>Demo with a browser login that shows a successful login into the application on the RP that is using the OP account.</p>

Table 49. Conclusion. Show where to find more information about OpenID Connect.

Scene	Audio	Onscreen Action
12	For more information, visit these online resources.	<p>Show information on documentation:</p> <p><b>WebSphere Liberty download page</b>  <a href="https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-environments/">https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-environments/</a></p> <p><b>OpenID Connect feature installation</b>            Server: <a href="https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectServer-1.0">https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectServer-1.0</a>            Client: <a href="https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectClient-1.0">https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.openidConnectClient-1.0</a></p> <p><b>IBM Knowledge Center - OpenID Connect main page</b>  <a href="http://www-01.ibm.com/support/knowledgecenter/api/content/n1/en-us/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_using_oidc.html">http://www-01.ibm.com/support/knowledgecenter/api/content/n1/en-us/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_using_oidc.html</a></p> <p><b>IBM DeveloperWorks (including OP/RP sample)</b>  <a href="http://www.ibm.com/developerworks/websphere/library/techarticles/1502_odonnell/1502_odonnell.html">http://www.ibm.com/developerworks/websphere/library/techarticles/1502_odonnell/1502_odonnell.html</a></p>

For more information about OpenID Connect, see [8.5.5.5](#) Using OpenID Connect.

## Video: Setting up Admin Center



8.5.5.2

The following transcript is for the “Setting up Admin Center” video, which demonstrates how to configure a `server.xml` file to enable Admin Center. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content shown in the video.

### Setting up Admin Center

Table 50. Demo setting up Admin Center in Liberty. Show `server.xml` changes.

Scene	Audio	Onscreen Action
1	This video shows how to enable Liberty Admin Center and get to the login page. Enabling Admin Center is quick and easy.	Show Liberty Admin Center login page in web browser.
2	You need a Liberty Version 8.5.5.2 or later installation. Make a few changes to the <code>server.xml</code> file of a server and you can use the Admin Center.	In Windows Explorer, show <code>wlp/usr</code> Liberty installation directory with no server defined.
3	I have just installed Liberty. To add a server, I go to a command prompt at the <code>wlp/bin</code> directory and run:  <code>server create myServer</code>  This creates a server named <code>myServer</code> .	Show command window open at <code>wlp/bin</code> , run <code>server create myServer</code> , and then show message <code>Server myServer created</code> .
4	The <code>server.xml</code> file is in the <code>usr/servers/myServer</code> directory. I want to make 4 changes to this <code>server.xml</code> file.	In Windows Explorer, select <code>wlp/usr/servers/myServer</code> directory. Select <code>server.xml</code> .
5	First, I add the <code>adminCenter-1.0</code> feature to the feature manager.	In editor open on <code>server.xml</code> , add the <code>adminCenter-1.0</code> feature to the feature manager:  <pre>&lt;featureManager&gt;   &lt;feature&gt;jsp-2.2&lt;/feature&gt;   &lt;feature&gt;adminCenter-1.0&lt;/feature&gt; &lt;/featureManager&gt;</pre>
6	Second, I add a <code>quickStartSecurity</code> element and specify a user name and password to configure a secure login. My user name is <code>admin</code> and my password is <code>adminpwd</code> .	In editor, add user name and password:  <pre>&lt;quickStartSecurity userName="admin"   userPassword="adminpwd" /&gt;</pre>
7	Third, I add a <code>keyStore</code> element to protect keystore files that have server authentication credentials. I choose <code>defaultKeyStore</code> for the <code>id</code> attribute and <code>Liberty</code> for the <code>password</code> attribute.	In editor, add keystore:  <pre>&lt;keyStore id="defaultKeyStore"   password="Liberty" /&gt;</pre>
8	Fourth, I add a <code>host</code> attribute set to asterisk to the <code>httpEndpoint</code> element. Setting <code>host</code> to asterisk, or to a defined host name, lets me view Admin Center on a cell phone, tablet, and remote computer, and not just on this <code>localhost</code> computer.	In editor, press enter after <code>id="defaultHttpEndpoint"</code> in the <code>httpEndpoint</code> element and add <code>host="*"</code> to the new line:  <pre>&lt;httpEndpoint id="defaultHttpEndpoint"   host="*"   httpPort="9080"   httpsPort="9443" /&gt;</pre>
9	Save the <code>server.xml</code> changes.	In editor, save changes.

Table 50. Demo setting up Admin Center in Liberty (continued). Show server.xml changes.

Scene	Audio	Onscreen Action
10	<p>Make these server.xml changes for every Liberty server and collective controller that you want to be able to view in Admin Center.</p> <p>In a multiple-server environment, you only need to make these changes to the server.xml file of a collective controller. You do not need to change the server.xml files of members.</p>	In editor, continue to show server.xml.

Table 51. Displaying Admin Center. Show starting the server, highlighting the adminCenter URL, and pointing a browser at the URL.

Scene	Audio	Onscreen Action
11	<p>I am now ready to start myServer and see Admin Center. Because I want messages shown in the foreground, I use the <b>run</b> command. I enter:</p> <pre>server run myServer</pre> <p>To not show the messages, I would enter:</p> <pre>server start myServer</pre>	In command window on wlp/bin directory, enter server run myServer. The command runs and product messages display in the command window.
12	Liberty writes status messages to the command window. One message has the URL for the adminCenter web application.	Highlight <code>http://host_name:9080/adminCenter/</code> URL in messages.
13	I put this URL into a web browser, and press Enter. The URL changes to https and port 9443 because I specified a secure connection in the server.xml file.	Add URL <code>http://host_name:9080/adminCenter/</code> to browser, press Enter, show URL change to <code>https://host_name:9443/adminCenter/</code> .
14	You might need to add a browser exception to confirm that the connection is trusted.	Show Firefox dialog and button selections <b>Add Exception</b> , <b>Get Certificate</b> , and then <b>Confirm Security Exception</b> .
15	I am now ready to log in to Admin Center.	Show Liberty Admin Center login page.

Table 52. Conclusion. Show where to find more information about Admin Center.

Scene	Audio	Onscreen Action
16	For information on logging into and using Admin Center, and viewing your personal Toolbox, see WASdev.net and the WebSphere Application Server Liberty documentation on IBM Knowledge Center.	<p>Show Liberty Admin Center login page with login fields and title covered by information on documentation:</p> <p><b>WASdev</b>  <a href="http://developer.ibm.com/wasdev">http://developer.ibm.com/wasdev</a></p> <p><b>WebSphere Application Server Liberty documentation on IBM Knowledge Center</b>  <a href="http://www-01.ibm.com/support/knowledgecenter/">http://www-01.ibm.com/support/knowledgecenter/</a></p> <p>Fade the information on documentation and end by showing just the Liberty Admin Center login page.</p>

For more information, see “Administering Liberty using Admin Center” on page 1074.

## Video: Thoughts on Liberty: Interview with Alasdair Nottingham

The following transcript is for the “Thoughts on Liberty: Interview with Alasdair Nottingham ” video. Alasdair shares his perspectives as a lead developer of WebSphere Application Server Liberty on why Liberty is exciting for developers. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

### Thoughts on Liberty: Interview with Alasdair Nottingham

*Table 53. Thoughts on Liberty: Interview with Alasdair Nottingham.* As lead developer for WebSphere Application Server Liberty, Alasdair Nottingham shares his perspectives on Liberty and its value for developers.

Scene	Audio	Onscreen Action
1	None	Show "IBM".
2	None	Show "Liberty".
3	None	Show the text:  Liberty is a new lightweight, easy to use version of WebSphere Application Server. We asked IBM developers to share their thoughts on what it means for developers ...
4	None	Show the question:  What do you find exciting about Liberty?
5	It's this really simple configuration that you can just go and edit with Notepad. I use Unix on a daily basis and being able to go in and edit it kinda completely breaks the expectation that people have of what WebSphere Application Server is. When we say, "Here is what Liberty is," the look of shock and disbelief of people seeing a runtime that starts in seconds, and is dynamic and flexible and has all the behaviors that Liberty has, I think is really exciting.	Show the developer, Alasdair Nottingham, talking on camera. Alasdair Nottingham is a lead developer for WebSphere Application Server Liberty.  Show words: <ul style="list-style-type: none"> <li>• Simple configuration</li> <li>• Breaks expectations</li> <li>• Dynamic</li> <li>• Flexible</li> </ul>
6	None	Show the question:  Tell us about a Liberty user experience?
7	We've got jazz.net which is a website hosting a whole lot of technologies around development. I got contacted one day by the jazz.net lead and he said, "I wanted you to that know we just got into production for our download application on jazz.net." He had not come and spoken to the development team or raised any issues. He had just gone ahead and taken an application that previously had been running elsewhere and got it up and running on jazz.net. All he was talking about what how great an experience it was. He said it was easy to tune and to get it to behave correctly. He was really excited.	Show the developer, Alasdair Nottingham, talking on camera.  Show words: <ul style="list-style-type: none"> <li>• Great experience</li> </ul>
8	None	Show the question:  What would you like to say to developers?

Table 53. *Thoughts on Liberty: Interview with Alasdair Nottingham (continued)*. As lead developer for WebSphere Application Server Liberty, Alasdair Nottingham shares his perspectives on Liberty and its value for developers.

Scene	Audio	Onscreen Action
9	<p>Well I think the important thing is to try it out. An awful lot of people have preconceptions on what proprietary software is like. Or what IBM software is like, in particular. I really think we have tried very hard to learn from the best examples of good usability.</p> <p>We've especially been focused around trying to make it good for developers. At the end of the day, I'm a developer. I work with a bunch of developers and we want to write something that is really good for developers to go and use. I think it really enables you to do a lot of thing that you couldn't do before.</p> <p>It runs in production just as easily. It's completely supported in production environments. That's something that people miss when we say it is a developer focused runtime. It's not to say we aren't trying to provide a runtime for production. It's just saying that by making sure we provide a great developer-friendly runtime, we get a great production-ready runtime at the same time.</p>	<p>Show the developer, Alasdair Nottingham, talking on camera.</p> <p>Show words:</p> <ul style="list-style-type: none"> <li>• Try it out</li> <li>• Good for developers</li> <li>• Completely supported in production</li> </ul>
10	None	<p>Show the text:</p> <p>To learn more: <a href="http://wasdev.net">http://wasdev.net</a></p>

## Video: Touring Admin Center



The following transcript is for the “Touring Admin Center” video, which briefly describes Admin Center features. This transcript is the video storyboard. The video does not have Audio text. Onscreen Action describes the images and words shown.

### Touring Admin Center

Table 54. *Touring Admin Center*. Show images of Admin Center and describe its features in on-screen text.

Scene	Audio	Onscreen Action
1	No narration. Background music.	Show title, <i>Touring Admin Center</i> .
2		Show words <i>Secure login</i> while showing the login page.
3		Show words <i>from a desktop, tablet and smartphone</i> while showing Admin Center login page in a web browser on a desktop monitor, then on a tablet, then on a smartphone.
4		Show words <i>Toolbox</i> and then <i>to hold tools and bookmarks of URLs to favorite websites</i> while showing the Toolbox.
5		Show words <i>customizable by each user</i> while showing Edit Toolbox page, Add Bookmark, and then Add Bookmark dialog.
6		Show words <i>Server Config tool</i> and then <i>to view and edit server.xml and configuration files</i> while showing Server Config tool open on a <i>server.xml</i> file.
7		Show words <i>Explore tool</i> and then <i>to browse applications, clusters, servers and hosts</i> while showing Explore tool Dashboard, then application details page, and then Servers page.

Table 54. *Touring Admin Center (continued)*. Show images of Admin Center and describe its features in on-screen text.

Scene	Audio	Onscreen Action
8		Show words <i>to manage resources</i> while showing Start action on two selected resources on Servers page and then search for resources on Search page.
9		Show words <i>to monitor resource metrics</i> while showing charts for a servlet on Monitor page.
10		Show words <i>and to set and view administrative metadata</i> while showing Tags and Metadata dialog and then metadata on server details page.
11		Show words <i>Deploy tool</i> and then <i>to install Liberty server packages on hosts in a collective</i> while showing Deploy tool.
12		Show words <i>Background Tasks</i> and then <i>to check the status of Deploy Installation tasks</i> and then <i>and to view messages</i> while showing Background Tasks page.
13		Show words <i>An interface that informs and guides</i> while showing message for Add Bookmark pop-up dialog.
14		Show words <i>and has preference options</i> while showing Preferences page.

Table 55. *Conclusion*. Show where to get Admin Center.

Scene	Audio	Onscreen Action
15	No narration. Background music.	Show words <i>Available from</i> and then <i>WASdev</i> and <i>http://developer.ibm.com/wasdev</i> .

For more information, see “Administering Liberty using Admin Center” on page 1074.

## Video: Using the IBM WebSphere Liberty Repository to enhance Liberty environments



The following transcript is for the “Using the IBM WebSphere Liberty Repository to enhance Liberty environments” video, which briefly describes the Liberty Repository, its rich set of assets, and how to obtain the assets. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content shown in the video.

### Using the Liberty Repository to enhance Liberty environments

Table 56. *Using the IBM WebSphere Liberty Repository to enhance Liberty environments*. Show images of WASdev.net and other methods of obtaining Liberty features, and describe the Liberty Repository in on-screen text.

Scene	Audio	Onscreen Action
1	Using the IBM WebSphere Liberty Repository to easily enhance your Liberty environment	Show title, <i>Using the IBM WebSphere Liberty Repository to easily enhance your Liberty environment</i> .
2	The rapid evolution and adoption of cloud, mobile, and social media technologies are driving the demand for delivering applications faster and more frequently. IBM WebSphere Application Server is now delivering features for Liberty on a continual basis via the Liberty Repository. In this video, you will learn about the Liberty Repository, its rich set of resources, and how to get started to obtain the latest Liberty features that are available.	Images representing, cloud, mobile, and social media appear. The words “develop applications faster,” “IBM WebSphere Application Server,” “New Liberty features on a continual basis,” and “Liberty Repository” are shown.

Table 56. Using the IBM WebSphere Liberty Repository to enhance Liberty environments (continued). Show images of WASdev.net and other methods of obtaining Liberty features, and describe the Liberty Repository in on-screen text.

Scene	Audio	Onscreen Action
3	<p>Building on the highly composable and modular nature of Liberty, we are making available new features on a continuous basis that you can use to easily extend or enhance your Liberty-based applications. The optional, production-ready features can be quickly and easily added to an existing WebSphere Liberty V8.5.5 installation. Simply choose the features that you want and then install the features to the applicable product service level. The features that you add inherit the same support of your existing installation.</p>	<p>Building blocks representing Liberty features drop down, building a stack of blocks. The word “Optional, production-ready features” and “Add to existing Liberty environment” are shown.</p>
4	<p>Some of the available features include:</p> <ul style="list-style-type: none"> <li>• IBM WebSphere Liberty Administrative Center</li> <li>• IBM WebSphere Liberty Connector Architecture</li> <li>• IBM WebSphere Liberty Optimized Adapters for z/OS</li> <li>• IBM WebSphere Liberty z/OS Connect</li> </ul>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• <i>IBM WebSphere Liberty Administrative Center</i></li> <li>• <i>IBM WebSphere Liberty Connector Architecture</i></li> <li>• <i>IBM WebSphere Liberty Optimized Adapters for z/OS</i></li> <li>• <i>IBM WebSphere Liberty z/OS Connect</i></li> </ul>
5	<p>With the Liberty Repository, you can easily obtain these composable features, as well as enhancements and helpful development tools, from the Repository, rather than having to wait for new product releases. The Liberty Repository enables us to deliver these valuable assets to you faster so that you can more quickly produce, enhance, and deliver innovations and engaging applications.</p>	<p>Show the WASdev Downloads page and the entries for features. Each entry includes an icon, the type of asset, the name of the asset, the release date, a description, and a rating.</p>
6	<p>In addition to features, the repository also includes artifacts such as administration scripts, samples, configuration snippets as well as artifacts that integrate open source projects more quickly and effectively. These assets are specifically designed to encompass end-to-end integration and provide important business value for the entire life-cycle of your Liberty application.</p>	<p>Show the WASdev Downloads page and the entries for other assets on the page. Each entry includes an icon, the type of asset, the name of the asset, the release date, a description, and a rating.</p> <p>Show a list of all possible asset types and their associated icons:</p> <ul style="list-style-type: none"> <li>• Features</li> <li>• Product samples</li> <li>• Addons</li> <li>• Product runtimes</li> <li>• Admin scripts</li> <li>• Config snippets</li> <li>• Open source integration</li> <li>• Tools</li> </ul>
7	<p>To get started, visit WASdev.net from your computer or mobile device. From the home page, click Downloads to browse and discover the variety of assets in the repository. You can search to find the assets you need. Also, you can use filtering to scope your search by asset type and edition. From WASdev.net, you can learn about the available features and how to install them using product installation tools, WebSphere Developer Tools or Rational® Application Developer.</p>	<p>Show the WASdev.net homepage. The cursor clicks on <b>Downloads</b> and scrolls through the asset listing. “MongoDB” is searched for, then the cursor clicks on the <b>Filter</b> menu and clicks <b>Feature</b> to filter the results to show only features.</p>

Table 56. Using the IBM WebSphere Liberty Repository to enhance Liberty environments (continued). Show images of WASdev.net and other methods of obtaining Liberty features, and describe the Liberty Repository in on-screen text.

Scene	Audio	Onscreen Action
8	The Liberty Repository has been seamlessly integrated with the Liberty installation methods. You can easily add Liberty features that are located in the Liberty Repository using either IBM Installation Manager, when installing Liberty, or using the command line, if Liberty is already installed. Both installation methods automatically search for and install any dependencies that the selected features might require.	Show the Installation Manager GUI window where additional features can be installed. Show a command-line window with the featureManager command, displaying options you can use with the command.
9	Additionally, the repository has been integrated with WebSphere Developer Tools and Rational Application Developer so that you can browse, filter, and search assets for these assets. After you find the asset that you want, simply add it to your install cart.	Show the Install Add-ons window in WebSphere Developer Tools. Click <b>Install</b> and hover over the Liberty Repository assets that can be installed.

Table 57. Conclusion. Show where to get started with Liberty Repository

Scene	Audio	Onscreen Action
10	To learn more about the Liberty Repository and the newly added features and assets, go to WASdev.net. Ready to get started?	Show words <i>WASdev</i> and <a href="https://www.ibmdev.net/wasdev/">https://www.ibmdev.net/wasdev/</a> .

For more information, see “Liberty Repository” on page 573.

## Video: Why Liberty? Performance that scales

8.5.5.6

The following transcript is for the “Why Liberty? Performance that scales” video, which describes how Liberty delivers optimal performance that easily scales. WebSphere Application Server Liberty is a lightweight, composable application server that is quick to start, easy to manage, and fast to deploy to, which enables rapid application development and availability in mobile, cloud, social, analytic, and enterprise production environments. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

### Why Liberty? Performance that scales

Table 58. Why Liberty? Performance that scales. Show title and then describe how the lightweight and composable nature of Liberty results in a runtime that delivers high performance and scalability.

Scene	Audio	Onscreen Action
1	Why Liberty? Performance that scales	Show title, <i>Why Liberty? Performance that scales</i> .
2	What if your runtime could help you build killer apps that scale?	Show words: What if your runtime could help you build killer applications that scale?



Table 58. Why Liberty? Performance that scales (continued). Show title and then describe how the lightweight and composable nature of Liberty results in a runtime that delivers high performance and scalability.

Scene	Audio	Onscreen Action
3	<p>WebSphere Application Server Liberty is the runtime for IBM Watson® as well as the hybrid cloud solution that serves every Grand Slam Tennis event around the world.</p> <p>So why do Watson™ and the tennis majors use Liberty? Because Liberty delivers the scalability and performance required to support the unique needs of cognitive and analytic queries. Liberty easily scales to meet the demands of real-time major sporting events and analysis applications that can quickly spike to over 100 million web page views at peak match times.</p> <p>Liberty is a lightweight, yet powerful, and fast runtime that was designed and built to meet the needs of developers. Using Liberty, you can rapidly develop and employ applications into production environments that perform and easily scale, which enables you to more quickly transform your own innovative ideas into reality!</p> <p>What if your runtime could help you build killer applications that scale, like Watson or the hybrid cloud solution for Grand Slam Tennis events? It can!</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Australian Open</li> <li>• French Open</li> <li>• Wimbledon</li> <li>• U.S. Open</li> <li>• Liberty can!</li> </ul> <p>Show graphics of:</p> <ul style="list-style-type: none"> <li>• IBM Watson</li> <li>• Liberty</li> <li>• Resizing square scaling graphic for <i>scalability</i></li> <li>• Lightning graphic for <i>performance</i></li> <li>• Multiple pop-up window graphics that show a tennis ball and racquet for <i>web page views</i></li> <li>• Scale graphic for <i>lightweight</i></li> <li>• Speedometer graphic for <i>3-second startup</i></li> <li>• Developer using Liberty</li> <li>• Arrow pointing at the developer with the word <i>You</i> above it</li> <li>• Thought bubble above developer for <i>innovative ideas</i></li> <li>• Thought bubble transferred to the developer's computer screen for <i>reality</i></li> </ul>
4	<p>Liberty is extremely lightweight. With a small memory footprint and fast application startup time, you can be up and running in a matter of seconds. In fact, Liberty is much lighter than the competition!</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Lightweight <ul style="list-style-type: none"> <li>– &lt; 65 MB footprint</li> <li>– &lt; 3-second start up</li> <li>– 40% lighter than the leading competitor</li> </ul> </li> </ul> <p>Show graphic of scale for <i>lightweight</i></p>
5	<p>The installation of Liberty is composable.</p> <p>For a fast startup time, the Liberty installation provides the essential features. If you need more features to meet your application needs, you can easily and quickly install them from the Liberty Repository.</p>	<p>Show words: Composable</p> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Show composable nature of Liberty with a foundation of blocks that is built up with more features, or blocks, added.</li> <li>• Show screen capture of Liberty repository and its assets.</li> </ul>

Table 58. Why Liberty? Performance that scales (continued). Show title and then describe how the lightweight and composable nature of Liberty results in a runtime that delivers high performance and scalability.

Scene	Audio	Onscreen Action
6	Liberty complies with the Java Platform Enterprise Edition (Java EE) 7 specification, for both the full platform and its subset, the Java EE 7 Web Profile. This support enables you to code without restrictions and use the latest Java EE 7 capabilities. You can pick and choose the Java EE 7 standard features that you want to use, or you can get all of the features in a convenience feature.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Fully Java EE 7 compliant               <ul style="list-style-type: none"> <li>– Full platform</li> <li>– Web Profile</li> <li>– Code without restrictions</li> <li>– Use the latest Java EE 7 capabilities</li> </ul> </li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Java logo</li> <li>• Rolling list of key supported Java EE 7 programming models.</li> <li>• List of Java EE 7 standard features</li> <li>• List of all Java EE 7 features</li> </ul>
7	Liberty can be deployed wherever, whenever and however. You have the flexibility to run in any environment and you can rapidly deploy to the cloud. This capability enables you to achieve even more performance in a short amount of time.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Whenever</li> <li>• Wherever</li> <li>• However</li> <li>• Rapidly deploy to cloud               <ul style="list-style-type: none"> <li>– On premises</li> <li>– Hosted</li> </ul> </li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Liberty graphic</li> <li>• Cloud with IBM Bluemix®</li> <li>• Growing box for <i>performance</i></li> <li>• Shrinking box for <i>time</i></li> </ul>
8	Finally, with the new auto scaling capabilities, you can rapidly scale the number of instances of the run time.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Scalable               <ul style="list-style-type: none"> <li>– 5 JVMs to 5,000 in minutes</li> </ul> </li> </ul> <p>Show graphic of square graphs quickly multiplying</p>
9	It's not enough to just build a great application. It needs powerful performance and scalability behind it. With these capabilities, Liberty gives you the performance that excels and scalability to meet user demands with fewer hardware and license requirements.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Lightweight</li> <li>• Composable</li> <li>• Java EE 7</li> <li>• Cloud ready</li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Resizing square scaling graphic for <i>scalability</i></li> <li>• Lightning graphic for <i>performance</i></li> <li>• Graph for <i>application</i></li> <li>• Shrinking graphics for <i>hardware and license requirements</i></li> </ul>

Table 58. *Why Liberty? Performance that scales (continued)*. Show title and then describe how the lightweight and composable nature of Liberty results in a runtime that delivers high performance and scalability.

Scene	Audio	Onscreen Action
10	There's no better time than now to get started with Liberty -- and it's easy! To download Liberty, and access a wide variety of features, samples, and other tips about developing and extending your applications, visit WASdev.net.	Show words: <ul style="list-style-type: none"> <li>• Getting started is easy</li> <li>• Download Liberty and get features, samples, development tips from WASdev.net</li> </ul>

## Video: Why Liberty? Fast application development

8.5.5.6

The following transcript is for the “Why Liberty? Fast application development” video, which describes how Liberty is simple to install and configure. WebSphere Application Server Liberty is a lightweight, composable application server that is quick to start, easy to manage, and fast to deploy to, which enables rapid application development and availability in mobile, cloud, social, analytic, and enterprise production environments. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

### Why Liberty? Fast application development

Table 59. *Why Liberty? Fast application development*. Show title and then describe how Liberty is simple to install and configure, and how it integrates with WebSphere Developer Tools.

Scene	Audio	Onscreen Action
1	Why Liberty? Fast application development	Show title, <i>Why Liberty? Fast application development</i> .
2	What if your runtime could help you develop apps faster?	Show words: What if your runtime could help you develop apps faster?
3	<p>Did you know you can gain about 8 weeks of time back per year by using WebSphere Application Server Liberty to develop your WebSphere applications?</p> <p>Liberty is simple to use, yet powerful, and was designed and built to meet the needs of developers. Using Liberty, you can rapidly develop and deploy your applications into production environments, which enables you to more quickly transform your own innovative ideas into reality!</p> <p>What if your runtime could give you back weeks each year? It can!</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Gain 8 weeks by using Liberty</li> <li>• Liberty can!</li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Calendar showing 8 weeks crossed off and now free</li> <li>• Code graphic for <i>simple</i></li> <li>• Lightning graphic for <i>powerful</i></li> <li>• Developer using Liberty</li> <li>• Arrow pointing at the developer with the word <i>You</i> above it</li> <li>• Thought bubble above developer for <i>innovative ideas</i></li> <li>• Thought bubble transferred to the developer's computer screen for <i>reality</i></li> </ul>

Table 59. Why Liberty? Fast application development (continued). Show title and then describe how Liberty is simple to install and configure, and how it integrates with WebSphere Developer Tools.

Scene	Audio	Onscreen Action
4	<p>Liberty is incredibly easy to use.</p> <p>With a footprint of less than 65 MB and a simple archive installation, Liberty provides a lightweight architecture to manage lightning fast, data-rich applications across any web, mobile, or cloud environment with high performance. After downloading Liberty, you can install and configure Liberty, add an application, and have that application running, all in one minute.</p> <p>Liberty has a simple configuration that you can easily share across your development team, store in version control, and edit in real time.</p> <p>When you upgrade your version of Liberty, you can continue to use existing applications and configurations with no changes. There's no need for migration!</p> <p>There are also zero restarts. After you save configuration changes, you do not need to restart Liberty to see the changes go live.</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Liberty is incredibly easy to use</li> <li>• Lightweight <ul style="list-style-type: none"> <li>– &lt; 65 MB footprint</li> <li>– Archive install</li> </ul> </li> <li>• Simple <ul style="list-style-type: none"> <li>– Simple install</li> <li>– Simple XML-based configuration</li> <li>– Zero migration</li> <li>– Zero restarts</li> </ul> </li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Cloud with Liberty logo sending thunder bolts to devices</li> <li>• For <i>simple install</i>, show install from .zip file</li> <li>• For <i>simple configuration</i>, show changing the server.xml file configuration from WebSphere Developer Tools</li> </ul>
5	<p>WebSphere Developer Tools make it easy to write and deploy applications in Eclipse. Simply drag your application onto your Liberty server.</p> <p>With the IBM Eclipse Tools for Bluemix, you can also quickly and easily deploy to Bluemix.</p>	<p>Show words: WebSphere Developer Tools</p> <p>Show screen capture of WebSphere Developer Tools and dragging a Liberty feature to install it</p>
7	<p>You can build great applications for mobile, cloud, social, analytic, and enterprise production environments without wasted time. Just imagine what you can do with extra weeks gained back each year!</p>	<ul style="list-style-type: none"> <li>• Show graphic of a happy developer</li> <li>• Show developer on vacation</li> </ul>
8	<p>There's no better time than now to get started with Liberty -- and it's easy! To download Liberty, and access a wide variety of features, samples, and other tips about developing and extending your applications, visit WASdev.net.</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Getting started is easy</li> <li>• Download Liberty and get features, samples, development tips from WASdev.net</li> </ul>

## Video: Why Liberty? Rapid deployment and powerful administration

8.5.5.6

The following transcript is for the “Why Liberty? Rapid deployment and powerful administration” video, which describes how Liberty provides a run time to deploy rapidly and easily manage your systems. WebSphere Application Server Liberty is a lightweight, composable application server that is quick to start, easy to manage, and fast to deploy to, which enables rapid application development and availability in mobile, cloud, social, analytic, and enterprise production environments. This transcript is the video storyboard. Audio describes narration and captions. Onscreen Action describes the content that is shown in the video.

 Why Liberty? Rapid deployment and powerful administration

Table 60. Why Liberty? Rapid deployment and powerful administration. Show title and then describe how Liberty is easy to manage and it integrates with open source frameworks.

Scene	Audio	Onscreen Action
1	Why Liberty? Rapid deployment and powerful administration	Show title, <i>Why Liberty? Rapid deployment and powerful administration.</i>
2	What if your runtime could speed up your application deployment?	Show words: What if your runtime could speed up your application deployment?
3	<p>Did you know that with WebSphere Application Server Liberty, you can deploy a large web application over 75% faster than the leading competitor? Not only does Liberty provide a single runtime that meets all of your Java EE needs, it also gives you rapid deployment and powerful administration capabilities.</p> <p>Liberty is a lightweight, but powerful and fast runtime that was designed and built to meet the needs of developers. Using Liberty, you can rapidly develop and deploy your applications into production environments, which enables you to more quickly transform your own innovative ideas into reality!</p> <p>What if your runtime could speed up your application deployment? It can!</p>	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Rapid deployment</li> <li>• Powerful administration</li> <li>• Liberty can!</li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Liberty logo</li> <li>• Speedometer graphic for <i>75% faster</i></li> <li>• Java EE logo</li> <li>• Scale graphic for <i>lightweight</i></li> <li>• Speedometer graphic for <i>fast</i></li> <li>• Developer using Liberty</li> <li>• Arrow pointing at the developer with the word <i>You</i> above it</li> <li>• Thought bubble above developer for <i>innovative ideas</i></li> <li>• Thought bubble transferred to the developer's computer screen for <i>reality</i></li> </ul>
4	Liberty is easy to manage. Your servers, applications, and other resources are right at your fingertips. With Admin Center, which is accessible from your smartphone, tablet, or computer, you can manage and monitor servers and applications, deploy server packages, and view bookmarked information.	<p>Show words: Liberty is incredibly easy to manage</p> <p>Show graphics for:</p> <ul style="list-style-type: none"> <li>• Show graphic screen captures of using Admin Center.</li> <li>• Show graphic of mentioned devices</li> </ul>

Table 60. Why Liberty? Rapid deployment and powerful administration (continued). Show title and then describe how Liberty is easy to manage and it integrates with open source frameworks.

Scene	Audio	Onscreen Action
5	Liberty integrates seamlessly with open source software. You can take advantage of Spring, Maven, Arquillian, MongoDB, and Docker, just to name a few. You don't need to reinvent the wheel or your existing applications. By leveraging Liberty's flexibility and integrating Liberty into your current development environment, you can be more productive and save valuable time.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Liberty integrates seamlessly with open source software</li> <li>• Open Source Integration</li> <li>• Software: <ul style="list-style-type: none"> <li>– Tapestry</li> <li>– MongoDB</li> <li>– Cassandra</li> <li>– And more!</li> </ul> </li> <li>• Frameworks: <ul style="list-style-type: none"> <li>– Arquillian</li> <li>– Docker</li> <li>– Spring</li> <li>– Chef</li> <li>– Puppet</li> <li>– And more!</li> </ul> </li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Wheel and existing applications</li> <li>• Check marks appearing under the graphics</li> </ul>
6	Liberty also supports continuous integration and DevOps. Liberty's fast start up time, small footprint, and simple configuration supports the continuous integration of code into frequent builds and automated testing. With simple build tools for packing the runtime into a single archive, Liberty is easy to deploy in production environments. Additionally, you can use the Liberty Chef cookbooks, which support the use of the Chef DevOps framework.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Liberty supports continuous integration and DevOps</li> <li>• Simple packaging tools</li> </ul> <p>Show graphics:</p> <ul style="list-style-type: none"> <li>• Apps and codes on a conveyer belt</li> <li>• Check marks above the apps and codes</li> <li>• Screen capture of packaging tools being used</li> <li>• Liberty Chef cookbook on Git</li> </ul>
7	It's not enough to just build a great application, you need to be able to deploy rapidly and easily manage your systems.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• It's not enough to just build a great application</li> <li>• Rapidly deployment</li> <li>• Powerful administration</li> </ul>
8	There's no better time than now to get started with Liberty -- and it's easy! To download Liberty, and access a wide variety of features, samples, and other tips about developing and extending your applications, visit WASdev.net.	<p>Show words:</p> <ul style="list-style-type: none"> <li>• Getting started is easy</li> <li>• Download Liberty and get features, samples, development tips from WASdev.net</li> </ul>

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Mail Station P300  
2455 South Road

Poughkeepsie, NY 12601-5400  
USA  
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

APACHE INFORMATION. The information center includes all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>. You may obtain a copy of the Apache License at <http://www.apache.org/licenses/LICENSE-2.0>.



## Programming interface information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of WebSphere Application Server. This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of WebSphere Application Server. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking: Programming Interface information.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

---

## Privacy Policy Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's privacy policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details/us/en> sections entitled "Cookies, Web Beacons and Other Technologies" and "Software Products and Software-as-a Service".



---

## Chapter 2. Migrating applications to Liberty

You can migrate applications to Liberty.

### Procedure

Migrate data access applications to the Liberty profile.

---

### Migrating data access applications to Liberty

For data access applications, you need to change configurations when you migrate a data source from the WebSphere Application Server traditional to Liberty.

### Procedure

- “Configuration differences between the traditional and Liberty: dataSource and jdbcDriver elements.”
- “Configuration differences between the traditional and Liberty: connectionManager element” on page 684.
- “Migrating a DB2 data source to Liberty” on page 685.
- “Migrating a Derby embedded data source to Liberty” on page 687.

### Configuration differences between the traditional and Liberty: dataSource and jdbcDriver elements

There are some differences in configuration between dataSource in Liberty and data sources in the traditional .

- Data source properties with different names
  - **ifxIFX\_LOCK\_MODE\_WAIT**, which is **informixLockModeWait** in traditional.
  - **supplementalJDBCTrace**, which is **supplementalTrace** in traditional.
  - **transactional**, which is **nonTransactionalDataSource** in traditional.
  - **isolationLevel**, which is **webSphereDefaultIsolationLevel** in traditional.
  - **queryTimeout**, which is **webSphereDefaultQueryTimeout** in traditional.
  - **id**, which is **name** in traditional.
- Data source properties with different values
  - **beginTranForResultSetScrollingAPIs**, which is true by default in Liberty
  - **beginTranForVendorAPIs**, which is true by default in Liberty
  - **connectionSharing**, which is MatchOriginalRequest by default in Liberty
  - **statementCacheSize**, which is is a JDBC provider property in traditional, and a **dataSource** property in Liberty, with a default value of 10.
- Data source properties in traditional that have no Liberty equivalent
  - **category**
  - **supportsDynamicUpdates**
- **connectionSharing** property of data sources
  - Liberty allows **connectionSharing** to be configured to either MatchOriginalRequest or MatchCurrentState. By default, it is **MatchOriginalRequest**.
  - The traditional allows **connectionSharing** to be configured in a finer grained manner, where individual connection properties can be matched based on the original connection request or the current state of the connection. In the traditional, **connectionSharing** is a combination of bits

representing which connection properties to match based on the current state of the connection. In the traditional, a value of 0 means to match all properties based on the original connection request; a value of -1 means to match all properties based on the current state of the connection. The default value for the traditional is 1, which means that the isolation level is matched based on the current state of the connection and all other properties are matched based on the original connection request.

- Time duration properties of data source

Time duration properties can optionally be specified with units in Liberty. For example,

```
<dataSource id="informix" jndiName="jdbc/informix" queryTimeout="5m" ...>
 <properties.informix ifxIFX_LOCK_MODE_WAIT="120s" .../>
</dataSource>
```

See **\*\*\*\* MISSING FILE \*\*\*\*** for accepted time units and formats of dataSource element. Omitting the units in Liberty is equivalent to the default units used in the traditional.

- Configuration for JDBC drivers

- In Liberty, you can take the same approach of configuring different jdbcDriver elements for XA capable and non-XA capable data source implementation classes. Alternatively, you can use a single jdbcDriver element for both. Defining multiple jdbcDriver elements does not cause different class loaders to be used. In Liberty, jdbcDriver elements always use the class loader of the shared library with which they are configured.
- In the traditional, a JDBC provider is defined to point to the JDBC driver JARs, compressed files, and native files. You must define separate JDBC providers for XA capable and non-XA capable data source implementation classes.

For some of the commonly used JDBC drivers, Liberty infers the data source implementation class names based on the names the driver JARs. Therefore, you can omit the implementation class names. For example:

```
<jdbcDriver id="Derby" libraryRef="DerbyLib"/>
<library id="DerbyLib">
 <fileset dir="C:/Drivers/derby" includes="derby.jar" />
</library>
```

Use the optional properties of the default implementation classes to override these classes such as javax.sql.DataSource, javax.sql.ConnectionPoolDataSource, and javax.sql.XADataSource.

The following example shows how to override the default javax.sql.XADataSource and javax.sql.ConnectionPoolDataSource implementations that Liberty selects

```
<jdbcDriver id="Derby" libraryRef="DerbyLib"
 javax.sql.XADataSource="org.apache.derby.jdbc.EmbeddedXADataSource"
 javax.sql.ConnectionPoolDataSource="org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource"/>
<library id="DerbyLib">
 <fileset dir="C:/Drivers/derby" includes="derby.jar" />
</library>
```

See **\*\*\*\* MISSING FILE \*\*\*\*** for more information about the jdbcDriver element.

## Configuration differences between the traditional and Liberty: connectionManager element

There are some differences in configuration between connectionManager in Liberty and connection pools in the traditional.

- Properties with different names
  - **maxConnectionsPerThread**, which is **maxNumberOfMCSAllowableInThread** in the traditional.
  - **maxIdleTime**, which is **unusedTimeout** in the traditional.
  - **maxPoolSize**, which is **maxConnections** in the traditional.
  - **minPoolSize**, which is **minConnections** in the traditional.

- Time duration properties

You can optionally specify the time duration properties with units in Liberty. For example,  
`<connectionManager id="pool1" connectionTimeout="30s" reapTime="3m" maxIdleTime="30m"/>`

See **\*\*\*\* MISSING FILE \*\*\*\*** for accepted time units and formats for the `connectionManager` element. If you do not specify time units in Liberty, the same default units are used as in the traditional.

- Differences between immediate timeout values and never (disable) timeout

There are differences in the values that represent immediate timeout and never (disabled) timeout.

- Liberty uses a value of 0 to represent immediate, whereas the traditional often uses -1 for immediate.
- Liberty uses a value of -1 to represent never (disabled), whereas the traditional often uses 0 for never (disabled).

Specifically this applies to the following attributes:

- **agedTimeout**
- **connectionTimeout**
- **maxIdleTime**, which is **unusedTimeout** in the traditional
- **reapTime**

- Purge policy changes

In Liberty, there are three purge policy values: **EntirePool**, **FailingConnectionOnly**, and **ValidateAllConnections**.

In the traditional, there are two purge policy values: **EntirePool** and **FailingConnectionOnly**, with a second property, `defaultPretestOptimizationOverride`, determining the behavior of **FailingConnectionOnly**.

Purge policies in Liberty, and their traditional equivalents, are as follows:

- `purgePolicy="EntirePool"`, which is the same for both.
- `purgePolicy="FailingConnectionOnly"`, which is equivalent to `purgePolicy="FailingConnectionOnly"` with `defaultPretestOptimizationOverride="false"` in the traditional.
- `purgePolicy="ValidateAllConnections"`, which is equivalent to `purgePolicy="FailingConnectionOnly"` with `defaultPretestOptimizationOverride="true"` in the traditional.

## Migrating a DB2 data source to Liberty

You can migrate a DB2 data source to Liberty.

### About this task

See the following code examples for the configurations for a DB2 data source in the traditional and Liberty.

### Example

In the traditional:

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1321914412932"
 providerType="DB2 Using IBM JCC Driver" isolatedClassLoader="false"
 implementationClassName="com.ibm.db2.jcc.DB2ConnectionPoolDataSource" xa="false">
 <classpath>${DB2_JCC_DRIVER_PATH}/db2jcc4.jar</classpath>
 <classpath>${DB2_JCC_DRIVER_PATH}/db2jcc_license_cu.jar</classpath>
 <classpath>${DB2_JCC_DRIVER_PATH}/db2jcc_license_cisuz.jar</classpath>
 <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1321914498985"
 name="DefaultDB2Datasource" jndiName="jdbc/DefaultDB2Datasource"
```

```

 providerType="DB2 Using IBM JCC Driver" authMechanismPreference="BASIC_PASSWORD"
 authDataAlias="IBM-9NE5C7ONIG4Node01/dbuser2" relationalResourceAdapter="builtin_rra"
 statementCacheSize="10"
 datasourceHelperClassname="com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper">
<propertySet xmi:id="J2EEResourcePropertySet_1321914499000">
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499000" name="databaseName"
 type="java.lang.String" value="TESTDB" required="true" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499001" name="driverType"
 type="java.lang.Integer" value="4" required="true" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499002" name="serverName"
 type="java.lang.String" value="localhost" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499003" name="portNumber"
 type="java.lang.Integer" value="50000" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499010" name="currentLockTimeout"
 type="java.lang.Integer" value="10" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499013" name="currentSchema"
 type="java.lang.String" value="DBUSER2" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499015" name="cursorSensitivity"
 type="java.lang.Integer" value="0" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499016" name="deferPrepares"
 type="java.lang.Boolean" value="true" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499027" name="loginTimeout"
 type="java.lang.Integer" value="0" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499032"
 name="resultSetHoldability" type="java.lang.Integer" value="1" required="false"
 ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499034"
 name="retrieveMessagesFromServerOnGetMessage"
 type="java.lang.Boolean" value="true" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499041" name="traceLevel"
 type="java.lang.Integer" value="-1" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499052"
 name="beginTranForResultSetScrollingAPIs" type="java.lang.Boolean" value="false"
 required="false" ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499053"
 name="beginTranForVendorAPIs" type="java.lang.Boolean" value="false" required="false"
 ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499054" name="connectionSharing"
 type="java.lang.Integer" value="-1" required="false" ignore="false"
 confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499060"
 name="nonTransactionalDataSource" type="java.lang.Boolean" value="false"
 required="false" ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499063"
 name="syncQueryTimeoutWithTransactionTimeout" type="java.lang.Boolean" value="false"
 required="false" ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499069"
 name="webSphereDefaultIsolationLevel" type="java.lang.Integer" value="2"
 required="false" ignore="false" confidential="false" supportsDynamicUpdates="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1321914499070"
 name="webSphereDefaultQueryTimeout" type="java.lang.Integer" value="10"
 required="false" ignore="false" confidential="false" supportsDynamicUpdates="false"/>

```

```

 </propertySet>
 <connectionPool xmi:id="ConnectionPool_1321914499012" connectionTimeout="180"
 maxConnections="10" minConnections="1" reapTime="180" unusedTimeout="1800"
 agedTimeout="7200" purgePolicy="EntirePool" />
 <mapping xmi:id="MappingModule_1321914681786" mappingConfigAlias=""
 authDataAlias="IBM-9NE5C7ONIG4Node01/dbuser2"/>
</factories>
</resources.jdbc:JDBCProvider>
<systemLoginConfig xmi:id="JAASConfiguration_2">
 <authDataEntries xmi:id="auth1" alias="IBM-9NE5C7ONIG4Node01/dbuser2"
 userId="dbuser2" password="{xor}LDcfLTo70z0=" />
</systemLoginConfig>

```

In Liberty, the equivalent configuration is:

```

<variable name="DB2_JCC_DRIVER_PATH" value="C:/Drivers/DB2" />
<library id="db2Lib">
 <fileset dir="{DB2_JCC_DRIVER_PATH}" includes="db2jcc4.jar
 db2jcc_license_cu.jar db2jcc_license_cisuz.jar" />
</library>
<dataSource id="DefaultDB2Datasource" jndiName="jdbc/DefaultDB2Datasource"
 statementCacheSize="10"
 beginTranForResultSetScrollingAPIs="false"
 beginTranForVendorAPIs="false"
 connectionSharing="MatchCurrentState"
 transactional="false"
 syncQueryTimeoutWithTransactionTimeout="false"
 isolationLevel="TRANSACTION_READ_COMMITTED"
 queryTimeout="10"
 >
 <jdbcDriver libraryRef="db2Lib"
 javax.sql.ConnectionPoolDataSource="com.ibm.db2.jcc.DB2ConnectionPoolDataSource"/>
 <properties.db2.jcc
 databaseName="TESTDB"
 driverType="4"
 serverName="localhost"
 portNumber="50000"
 currentLockTimeout="10"
 currentSchema="DBUSER2"
 cursorSensitivity="0"
 deferPrepares="true"
 loginTimeout="0"
 resultSetHoldability="1"
 retrieveMessagesFromServerOnGetMessage="true"
 traceLevel="-1"
 user="dbuser2"
 password="{xor}LDcfLTo70z0="
 />
 <connectionManager connectionTimeout="180" maxPoolSize="10" minPoolSize="1" reapTime="180"
 maxIdleTime="1800" agedTimeout="7200" purgePolicy="EntirePool"/>
</dataSource>

```

## Migrating a Derby embedded data source to Liberty

You can migrate a Derby Embedded data source to Liberty.

### About this task

See the following code examples for the configurations for a Derby Embedded data source in the traditional and Liberty.

## Example

In the traditional:

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1183122153343"
 providerType="Derby JDBC Provider"
 implementationClassName="org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource"
 xa="false">
 <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</classpath>
 <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1183122153625"
 name="DefaultDerbyDatasource" jndiName="jdbc/DefaultDatasource"
 providerType="Derby JDBC Provider" authMechanismPreference="BASIC_PASSWORD"
 relationalResourceAdapter="builtin_rra" statementCacheSize="10"
 datasourceHelperClassName="com.ibm.websphere.rsadapter.DerbyDataStoreHelper">
 <propertySet xmi:id="J2EEResourcePropertySet_1183122153625">
 <resourceProperties xmi:id="J2EEResourceProperty_1183122153625" name="databaseName"
 type="java.lang.String" value="C:/myDerby/DefaultDB" required="true"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1183122153626" name="shutdownDatabase"
 type="java.lang.String" value="false" required="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1183122153629" name="connectionAttributes"
 type="java.lang.String" value="upgrade=true" required="false"/>
 <resourceProperties xmi:id="J2EEResourceProperty_1183122153630" name="createDatabase"
 type="java.lang.String" value="create" required="false"/>
 </propertySet>
 <connectionPool xmi:id="ConnectionPool_1183122153625" connectionTimeout="180"
 maxConnections="10" minConnections="1" reapTime="180" unusedTimeout="1800"
 agedTimeout="7200" purgePolicy="EntirePool"/>
 </factories>
</resources.jdbc:JDBCProvider>
```

In Liberty, the equivalent configuration is:

```
<variable name="DERBY_JDBC_DRIVER_PATH" value="C:/Drivers/derby" />
<library id="derbyLib">
 <fileset dir="${DERBY_JDBC_DRIVER_PATH}" includes="derby.jar" />
</library>
<dataSource id="DefaultDerbyDatasource" jndiName="jdbc/DefaultDatasource"
 statementCacheSize="10">
 <jdbcDriver libraryRef="derbyLib"
 javax.sql.ConnectionPoolDataSource="org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource"/>
 <properties.derby.embedded
 databaseName="C:/myDerby/DefaultDB"
 shutdownDatabase="false"
 connectionAttributes="upgrade=true"
 createDatabase="create"
 />
 <connectionManager connectionTimeout="180" maxPoolSize="10" minPoolSize="1" reapTime="180"
 maxIdleTime="1800" agedTimeout="7200" purgePolicy="EntirePool" />
</dataSource>
```



---

## Chapter 3. Installing Liberty

There are two methods for installing the WebSphere Application Server Liberty Core. You can use Installation Manager or use downloaded archive files. With either method, you can also download additional assets from the Liberty Repository.

### About this task

- | **New:** Liberty now follows a continuous delivery process with a new fix pack numbering scheme. Fix pack 16.0.0.2 is the next fix pack after 8.5.5.9. For more information about fix pack 16.0.0.2, see [What is new in Liberty](#) in the new location of the latest Liberty documentation.

When choosing installation methods, consider the packaging strategy of each method.

- For Installation Manager, you can download a single package that contains all requested content.
- For downloaded archive files, content is packaged in multiple archives to reduce download size.

The following table compares the available installation methods for WebSphere Application Server Liberty Core.

*Table 61. Comparison of available installation methods*

	Installation Manager	Archive
Agent-less installation	No	Yes
Install directly to any fix pack level	Yes	Yes
Apply fix packs in place	Yes	No
Apply interim fixes in place	Yes	Yes
Automated rollback of fix packs and interim fixes	Yes	No
Upgrade product editions	Yes	8.5.5.5 Yes
Pluggable Java SDK provided	Yes	No
Integrated with developer tools	No	Yes
Add features in place	Yes	Yes
Liberty Repository integration	Yes	Yes
z/OS packaging provided	Yes	No
Minification supported	Yes, but the output of the minify command is a new image that can only be serviced with archive procedures.	Yes, the output of the minify command is a new image.

### Procedure

1. Install Liberty using one of the following methods:
  - a. Install Liberty using the Installation Manager.
  - b. Install Liberty using downloaded files and archives. You can install Liberty from a self-extracting Java archive file or a ZIP archive file.
2. Optional: Install assets from the Liberty Repository.

---

## Installing and uninstalling Liberty using Installation Manager

You can install WebSphere Application Server Liberty using IBM Installation Manager.

### About this task

**Note:** 8.5.5.11 Support for using Java SE 6 with WebSphere Liberty ends in September 2017. After the end of support, the Liberty kernel will be recompiled and can no longer run with Java SE 6. If you continue to use Java SE 6 on earlier fix packs after the end of support date, you could expose your environment to security risks.

Java SE 8 is the recommended Java SDK because it provides the latest features and security updates. You can install it by installing the IBM SDK, Java Technology Edition, Version 8 package to the package group that contains WebSphere Liberty.

Installation Manager is a general-purpose software installation and update tool that runs on a range of computer systems.

**Note:** Although WebSphere Application Server Liberty can be installed and maintained using Installation Manager Version 1.5.2 and later, this information is optimized for use with Installation Manager Version 1.6.2 and later. To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

**Note:** Starting with Version 1.6.2, Installation Manager supports Mac OS X. Installing and maintaining a Liberty offering using Installation Manager on Mac OS X is supported for the following offerings:

- IBM WebSphere Application Server Liberty
- IBM WebSphere Application Server Liberty Trial
- IBM WebSphere Application Server Liberty Core
- IBM WebSphere Application Server Liberty Core Trial
- IBM WebSphere Application Server Liberty - Express
- IBM WebSphere Application Server Liberty Network Deployment
- IBM WebSphere Application Server Liberty Network Deployment Trial
- IBM WebSphere Application Server Liberty for Developers
- IBM WebSphere Application Server Liberty for Developers (ILAN)

Installation Manager for Mac OS X is not contained in the product image or on the product media. You can obtain Installation Manager Version 1.6.2 and later from the IBM Installation Manager download website. Refer to the System Requirements for IBM Installation Manager website and System Requirements for WebSphere Application Server Version 8.5.5 website for complete details on Mac OS X support.

| **Note:** You can install WebSphere Application Server Liberty using the Installation Manager Graphical User Interface (GUI) or Installation Manager in console mode. Console mode is an interactive text-based user interface to Installation Manager. Use console mode when you do not have a graphics display device available or when you want to run the Installation Manager without the graphical user interface. For example, use console mode for server-side deployments when no graphical user interface is present, or for running the installation from a remote host.

| **New:** Liberty now follows a continuous delivery process with a new fix pack numbering scheme. Fix pack 16.0.0.2 is the next fix pack after 8.5.5.9. You can continue to use the same Version 8.5 Installation Manager repositories and offering IDs to install or update to 16.0.0.2, or you can use the new versionless repositories and offerings. For more information about fix pack 16.0.0.2, see What is new in Liberty in the new location of the latest Liberty documentation.

Choose from the following options for more information on using Installation Manager for your installation.

## Procedure

- To install Liberty using Installation Manager, choose from the following options:
  - **Distributed operating systems** “Installing and uninstalling Liberty on distributed operating systems”
  - **IBM i** “Installing and uninstalling Liberty on IBM i operating systems” on page 788
- **8.5.5.1** If you want to use Installation Manager in console mode, see:
  - Installation Manager Version 1.6 documentation or the Installation Manager Version 1.5 documentation.

## Installing and uninstalling Liberty on distributed operating systems

IBM Installation Manager is a common installer for many IBM software products. You can use Installation Manager to install and manage the product lifecycle of WebSphere Application Server Liberty Core.

### Before you begin

Installation Manager is a single installation program that can use remote or local software flat-file repositories to install, modify, or update WebSphere Application Server products. It determines and shows available packages - including products, fix packs, interim fixes, and so on - checks prerequisites and interdependencies, and installs the selected packages. You also use Installation Manager to easily uninstall the packages that it installed.

#### Restrictions:

- If you have an earlier Alpha or a Beta version of WebSphere Application Server Liberty Core installed, uninstall it before installing this version.
- **Windows** If a non-administrator installs WebSphere Application Server Liberty Core on a Windows Vista, Windows 7, or Windows Server 2008 operating system into the Program Files or Program Files (x86) directory with User Account Control (UAC) enabled, WebSphere Application Server Liberty Core will not function correctly.

UAC is an access-control mechanism that allows non-administrative users to install a software product into the Program Files or Program Files (x86) directory; but it then prohibits any write access to that directory after the installation has completed.

To resolve this issue, perform one of the following actions:

- Install the offering into a directory other than Program Files or Program Files (x86).

For example:

C:\IBM\WebSphere\Liberty

- Disable UAC.

- When you install an offering using Installation Manager with local repositories, the installation takes a significantly longer amount of time if you use a compressed repository file directly without extracting it.

Before you install an offering using local repositories, extract the compressed repository file to a location on your local system before using Installation Manager to access it.

**Important:** Do not transfer the content of a repository in non-binary mode and do not convert any content on extraction.

**Tip:** Although almost all of the instructions in this section of the documentation work with earlier versions of IBM Installation Manager, this information is optimized for users who installed or upgraded to Installation Manager Version 1.6.2 or later. Version 8.5.5.4 and later of Liberty require Installation Manager Version 1.6.2 or later.

- | **Tip:** Different users can use WebSphere Liberty by using two different methods.
- | 1. Install a new WebSphere Liberty instance for each user. Each WebSphere Liberty install is a new user profile.
- | 2. Create multiple servers with different users. Each user should be part of a group that has access to the wlpdirectory and java\_home used.
- | If you use option two, run the command to create the server as the user who will run the server and create the server in the user's home directory. If you are using Linux, the command is similar to `su user1 export WLP_USER_DIR=/home/user1 server create Server1`.
- | Setting WLP\_USER\_DIR in the user's shell profile makes it easy to ensure that all Liberty commands act on the correct user directory.

**Important:** Installation Manager can install any fix-pack level of the offering directly without installing the intermediate fix packs; in fact, Installation Manager installs the latest level by default. For example, you can skip fix-pack levels and go from Version 8.5.5.1 directly to Version 8.5.5.5. Keep in mind, however, that later you can not roll back to any level that was skipped. If you directly install to Version 8.5.5.5, for example, you cannot roll back to Version 8.5.5.4. If you skip from Version 8.5.5.1 to Version 8.5.5.5, you can only roll back to Version 8.5.5.1. You should plan your installations accordingly.

## About this task

Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.

Perform one of these procedures to install or uninstall WebSphere Application Server Liberty Core using Installation Manager.

## Procedure

- “Installing Liberty on distributed operating systems using the GUI” on page 697
- “Installing Liberty on distributed operating systems by using the command line” on page 700
- “Installing Liberty on distributed operating systems by using response files” on page 703
- “Uninstalling Liberty from distributed operating systems using the GUI” on page 709
- “Uninstalling Liberty from distributed operating systems using the command line” on page 709
- “Uninstalling Liberty from distributed operating systems by using response files” on page 710

## Results

### Notes on logging and tracing:

- An easy way to view the logs is to open Installation Manager and go to **File > View Log**. An individual log file can be opened by selecting it in the table and then clicking the **Open log file** icon.
- Logs are located in the logs directory of Installation Manager's application data location. For example:

– **Windows** **Administrative installation:**

C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager

– **Windows** **Non-administrative installation:**

C:\Documents and Settings\user\_name\Application Data\IBM\Installation Manager

– **AIX** **HP-UX** **Linux** **Solaris** **Administrative installation:**

/var/IBM/InstallationManager

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrative installation:**

user\_home/var/ibm/InstallationManager

- The main log files are time-stamped XML files in the logs directory, and they can be viewed using any standard web browser.
- The log.properties file in the logs directory specifies the level of logging or tracing that Installation Manager uses.

### Notes on troubleshooting:

- **HP-UX** When you attempt to launch Installation Manager from a DVD that was mounted using the CD-ROM file system (CDFFS) on an HP-UX operating system, it might fail to launch and point to a log file that contains an exceptions similar to one of the following:

```
java.util.zip.ZipException: Exception in opening zip file:
org.osgi.framework.BundleException: Exception in
org.eclipse.update.internal.configurator.ConfigurationActivator.start()
or bundle org.eclipse.update.configurator.
```

This issue might be caused by Installation Manager reaching the upper limit of number of descriptors that can be opened on a CDFFS-mounted device. This upper limit is determined by the value for the tunable kernel parameter ncdnode, which specifies the maximum number of CDFFS nodes that can be in memory at any given time. To resolve the problem, change the ncdnode system kernel setting to 250. If the problem persists, increase the setting.

- **HP-UX** By default, some HP-UX systems are configured to not use DNS to resolve host names. This could result in Installation Manager not being able to connect to an external repository.

You can ping the repository, but nslookup does not return anything.

Work with your system administrator to configure your machine to use DNS, or use the IP address of the repository.

- In some cases, you might need to bypass existing checking mechanisms in Installation Manager.
  - On some network file systems, disk space might not be reported correctly at times; and you might need to bypass disk-space checking and proceed with your installation. To bypass disk-space checking, add `cic.override.disk.space=true` to the `config.ini` file in `IM_install_root/eclipse/configuration` and restart Installation Manager.
  - To bypass operating-system prerequisite checking, add `disableOSPreqChecking=true` to the `config.ini` file in `IM_install_root/eclipse/configuration` and restart Installation Manager.

If you need to use any of these bypass methods, contact IBM Support for assistance in developing a solution that does not involve bypassing the Installation Manager checking mechanisms.

- For more information on using Installation Manager, read the IBM Installation Manager Information Center.

Read the release notes to learn more about the latest version of Installation Manager. To access the release notes, complete the following task:

- **Windows** Click **Start > Programs > IBM Installation Manager > Release Notes**.
- **AIX** **HP-UX** **Linux** **Solaris** Go to the documentation subdirectory in the directory where Installation Manager is installed, and open the `readme.html` file.
- If a fatal error occurs when you try to install the offering, take the following steps:
  - Make a backup copy of your current installation directory in case IBM support needs to review it later.
  - Use Installation Manager to uninstall everything that you have installed under the installation location (package group). You might run into errors, but they can be safely ignored.
  - Delete everything that remains in the installation directory.
  - Use Installation Manager to reinstall the offering to the same location or to a new one.

- **8.5.5.1** If you are installing WebSphere Application Server Liberty Version 8.5.5.1 in console mode, you might receive warning messages similar to the following example:

No "conClass" attribute in "commonPanel" element of panel com.ibm.was.liberty.userdata.panel.UserData in com.ibm.was.liberty.userdata.panel.

This problem is caused by an IBM Installation Manager API which is deprecated in Version 1.6.2. These warning messages can be ignored. No action is required.

## Installing Installation Manager and preparing to install Liberty

Install Installation Manager and obtain the necessary product repositories before installing WebSphere Application Server Liberty Core.

### About this task

To install Liberty, you must have Installation Manager Version 1.5.2 or later.

**Note:** **8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

#### 1. Install Installation Manager.

- a. Obtain the necessary installation files.

There are three basic options for obtaining Installation Manager Version 1.5.2 or later.

- **Access the physical media.**
- **Download the files from the Passport Advantage site.**

Licensed customers with a Passport Advantage ID and password can download Installation Manager from the Passport Advantage site.

**Tip:** See How to download WebSphere Application Server V8.5.5 from Passport Advantage Online.

- **Download the most current version of Installation Manager from the download website.**  
You can download the most current version of Installation Manager from the IBM Installation Manager download website.

- b. Change to the location containing the Installation Manager installation files, and run one of the following commands.

#### Administrative installation:

- **Windows** install.exe
- **AIX** **HP-UX** **Linux** **Solaris** ./install

#### Non-administrative installation:

- **Windows** userinst.exe
- **AIX** **HP-UX** **Linux** **Solaris** ./userinst

#### Group-mode installation:

- **AIX** **HP-UX** **Linux** **Solaris** ./groupinst

#### Notes® on group mode:

- Group mode allows multiple users to use a single instance of IBM Installation Manager to manage software packages.  
This does not mean that two people can use the single instance of IBM Installation Manager at the same time.
- **Windows** Group mode is not available on Windows operating systems.

- If you do not install Installation Manager using group mode, you will not be able to use group mode to manage any of the products that you install later using this Installation Manager.
- Make sure that you change the installation location from the default location in the current user's home directory to a location that is accessible by all users in the group.
- Set up your groups, permissions, and environment variables as described in the Group mode road maps in the IBM Installation Manager documentation before installing in group mode.
- For more information on using group mode, read the Group mode road maps in the IBM Installation Manager documentation.

The installer opens an **Install Packages** window.

c. Make sure that the Installation Manager package is selected, and click **Next**.

d. Accept the terms in the license agreements, and click **Next**.

The program creates the directory for your installation.

e. Click **Next**.

f. Review the summary information, and click **Install**.

- If the installation is successful, the program displays a message indicating that installation is successful.
- If the installation is not successful, click **View Log File** to troubleshoot the problem.

## 2. Update an Installation Manager installation that is already on your system.

For information on updating Installation Manager to Version 1.5.2 or later, see the IBM Installation Manager documentation.

## 3. Obtain the product repositories.

There are three basic options for accessing the product repositories to install the offering.

### • Access the physical media, and use local installation.

You can access the product repositories on the media.

### • Download the files from the Passport Advantage site, and use local installation.

Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

**Tip:** See How to download WebSphere Application Server V8.5.5 from Passport Advantage Online for a list of the IBM WebSphere Application Server Liberty Core installation images downloadable from the IBM Passport Advantage Online website and other information.

### • Access the live repositories, and use web-based installation.

If you have a Passport Advantage ID and password, you can access the product repositories and install the offering from the web-based repositories. Use Installation Manager to install the offering from the web-based repository located at

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85>

**Note:** This location does not contain a web page that you can access using a web browser. This is a remote web-based repository location that you must add to your Installation Manager preferences before the Installation Manager GUI can access the files in this repository to install the offering.

**Tip:** This live repository is accessed by using Passport Advantage authentication. After you have installed Installation Manager, you can set the Passport Advantage preference to connect to the live repositories. To set Passport Advantage preferences, follow this procedure:

- Open Installation Manager.
- Open the Passport Advantage preferences page by selecting **File > Preferences > Passport Advantage**.
- Select **Connect to Passport Advantage** to connect to the Passport Advantage repository.

The Password Required dialog box opens.



- d. Enter a user name and password for Passport Advantage.
- e. **Optional:** Select **Save password** to save the user name and password credentials.  
If you do not save the user name and password credentials, you are prompted for these credentials each time you access Passport Advantage.
- f. Click **OK** to close the Password Required dialog box.
- g. Click **OK** to close the Preferences window.

For more information on setting your Installation Manager preferences, see the IBM Installation Manager documentation.

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

**Notes:**

- If you do not have a Passport Advantage ID and password, you must install the offering from the product repositories on the media or local repositories.
- With the Packaging Utility, you can create and manage packages for installation repositories. You can copy multiple packages into one repository or copy multiple disks for one product into a repository. You can copy packages from Passport Advantage or a web-based repository into a local repository for example. For more information on the Packaging Utility, go to the IBM Installation Manager documentation.

4.   **Optional:** Configure an instance of the Liberty Asset Repository Service or a local directory-based repository.

As part of installing Liberty, you can choose to install assets from the following repositories:






- The IBM WebSphere Liberty Repository, a public IBM-hosted repository that is accessible through the internet. For more information, see “Liberty Repository” on page 573.
- The Liberty Asset Repository Service, an open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. For more information, see the WASdev/tool.lars project on GitHub.
- Local directory-based repositories that are created by using the **installUtility download** action. For more information, see “Downloading assets using the installUtility command” on page 860.

Access to the IBM WebSphere Liberty Repository is enabled by default and requires internet access. If your system does not have internet access or you want to install customized Liberty assets, set up an instance of the Liberty Asset Repository Service or a local directory-based repository, and then add the repository in Installation Manager. For more information about the Liberty repositories, see “Installing assets using Installation Manager” on page 872.

5. **Optional: If you will be using the Installation Manager GUI, add repositories to your Installation Manager preferences.**

- a. Launch Installation Manager.
- b. Click **File > Preferences**.
- c. Select **Repositories**.
- d. Perform the following actions:
  - 1) Click **Add Repository**.
  - 2) Enter the path to the repository.config file in the location containing the repository files.

For example:

-  C:\repositories\offering\_name\local-repositories
-     /var/repositories/offering\_name/local-repositories

or



3) Click **OK**.

- e. Deselect any locations listed in the Repositories window that you will not be using.
- f. Click **Apply**.
- g. Click **OK**.
- h. Click **File > Exit** to close Installation Manager.

For more information on setting your Installation Manager preferences, see the IBM Installation Manager documentation.

#### 6. Optional: If you will be using the Installation Manager GUI, set your Installation Manager rollback preferences.

By default, Installation Manager saves earlier versions of a package to roll back to if you experience issues later. When Installation Manager rolls back a package to a previous version, the current version of the files are uninstalled and the earlier versions are reinstalled. If you choose not to save the files for rollback, you can prevent the files from being saved or delete them after they are saved. To set your rollback preferences, perform the following actions before installing a package:

- a. Launch Installation Manager.
- b. Open the Rollback preferences window by selecting **File > Preferences > Files for Rollback**.
- c. Select or clear the **Save files for rollback** option to save or to stop saving a copy of files that are required to roll back packages on your computer.

You can remove any files that have already been saved by clicking **Delete Saved Files**. If you delete the files and you need to roll back a package later, you must connect to a repository or insert the media to obtain the required files for the previous version of the package.

- d. Click **OK** to save your rollback preferences.

For more information on setting your Installation Manager preferences, see the IBM Installation Manager documentation.

## What to do next

Use Installation Manager to install the offering.

- “Installing Liberty on distributed operating systems using the GUI”
- “Installing Liberty on distributed operating systems by using the command line” on page 700
- “Installing Liberty on distributed operating systems by using response files” on page 703

## Installing Liberty on distributed operating systems using the GUI

You can use the Installation Manager GUI to install WebSphere Application Server Liberty Core.

### Before you begin

Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.

### About this task

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

1. Start Installation Manager.

**Tip:** AIX HP-UX Linux Solaris You can start Installation Manager in group mode with the `./IBMIM` command.

- Group mode allows multiple users to use a single instance of IBM Installation Manager to manage software packages.
- For more information on using group mode, read the Group mode road maps in the IBM Installation Manager documentation.

2. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

3. Perform the following actions.

- a. Select **IBM WebSphere Application Server Liberty Core** and the appropriate version.

**Note:** If you are installing the trial version of this product, select the appropriate trial edition and the appropriate version.

If you already have the WebSphere Application Server Liberty Core offering installed on your system in the installation location, a message displays indicating that the product is already installed. To create another installation of the offering in another location, click **Continue**.

**Tip:** If the **Search service repositories during installation and updates** option is selected on the Installation Manager Repository preference page and you are connected to the Internet, you can click **Check for Other Versions and Extensions** to search for updates in the default update repositories for the selected packages. In this case, you do not need to add the specific service-repository URL to the Installation Manager Repository preference page.

- b. Select the fixes to install.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

- c. Click **Next**.

**Note:** Installation Manager might prompt you to update to the latest level of Installation Manager when it connects to the repository. Update to the newer version before you continue if you are prompted to do so. Read the IBM Installation Manager documentation for information about automatic updates.

4. Accept the terms in the license agreements, and click **Next**.

5. Specify the installation root directory for the product binaries, which are also referred to as the core product files or system files.

The panel also displays the shared resources directory and disk-space information.

**Note:** The first time that you install a package using Installation Manager, you can specify the shared resources directory. The shared resources directory is where installation artifacts are located that can be used by one or more package groups. It is also used as a staging area for the product payload during installation operations. By default, this content is cached so that it can be used for rollback. Consider setting your rollback preferences to save these files as described in “Installing Installation Manager and preparing to install Liberty” on page 694. Use your largest drive for this installation. You cannot change the directory location until after you uninstall all packages.

If you are installing on a 64-bit system, use the bit-selection option to select the bitness of the installed Java libraries. If you are installing on a 32-bit system, this option is not available. When installing on a 64-bit system, you must select either the 32-bit or 64-bit option. This choice will determine the architecture bitness of the installation profile group. All extension offerings to WebSphere Application Server Liberty Core will be installed under the same profile group and have the same architecture bitness. For example, WebSphere SDK Java Technology Edition Version 7.0 for

Liberty is an extension offering to WebSphere Application Server Liberty Core and its installation will use a 32-bit or 64-bit JDK depending on your choice here.

**Restrictions:**

- Deleting the default target location and leaving an installation-directory field empty prevents you from continuing.
- Do not use symbolic links as the destination directory.  
Symbolic links are not supported.
- Do not use a semicolon in the directory name.  
WebSphere Application Server Liberty Core cannot install properly if the target directory includes a semicolon.

**Windows** A semicolon is the character used to construct the class path on Windows systems.

6. Click **Next**.

7. Select the features that you want to install.

Choose from the following features:

- IBM WebSphere Application Server Liberty Core
  - Embeddable EJB container and JPA client


This option installs the embeddable EJB container and JPA client.

The embeddable EJB container is a Java Archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

**Tip:** You can run Installation Manager later to modify this installation and add or remove this feature.

8. Click **Next**.

9.  **8.5.5.2** Optional: Install additional Liberty Repository assets. If you do not want to select any additional assets, you can skip this step. To install Liberty Repository assets, you must have IBM Installation Manager Version 1.6.2 or later. To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

a. Select whether you want to install assets from the Liberty Repository, and click **Next**. To install assets from the IBM WebSphere Liberty Repository, you must have access to the internet.

**8.5.5.6** If you choose not to connect to the IBM WebSphere Liberty Repository, you can still install assets from configured directory-based repositories or an instance of the Liberty Asset Repository Service. For more information, see “Installing assets using Installation Manager” on page 872.

b. **8.5.5.4** Click **Launch Asset Selection Wizard**.

c. You can perform a case-insensitive search for assets by display name or description. If you search with the **Keyword** field empty, then the search displays all applicable assets.

**8.5.5.5** With Version 8.5.5.5 and later, you can also search for assets by short name.

d. Select each asset that you want to install, then click **Next**.

e. Accept the license agreement and click **Finish**.

10. Click **Next**.

11. Review the summary information, and click **Install**.

- If the installation is successful, the program displays a message indicating that installation is successful. The program might also display important post-installation instructions.

- If the installation is not successful, click **View Log File** to troubleshoot the problem.
12. Click **Finish**.
  13. Click **File > Exit** to close Installation Manager.

## Installing Liberty on distributed operating systems by using the command line

You can install WebSphere Application Server Liberty Core by using the Installation Manager command line.

### Before you begin

**Important:** Before you install WebSphere Application Server Liberty Core, you must read the license agreement that you can find with the product files. Signify your acceptance of the license agreement by specifying `-acceptLicense` in the command as described in this topic.

Prepare the system onto which you want to install WebSphere Application Server Liberty Core as described in “Installing Installation Manager and preparing to install Liberty” on page 694.

### About this task

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

1. Optional: If the repository requires a user name and password, create credential-storage and master-password files to access this repository.

**Tip:** When you create a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Log on to your system.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
4. Verify that the offering repository is available.

Windows

```
imcl.exe listAvailablePackages -repositories source_repository
```

AIX

HP-UX

Linux

Solaris

```
./imcl listAvailablePackages -repositories source_repository
```

You see one or more levels of the offering.

5. Use the `imcl` command to install the offering.

Windows

```
imcl.exe install com.ibm.websphere.liberty.v85_offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory installation_directory
-sharedResourcesDirectory shared_directory
```

```
-preferences preference_key=value
-properties property_key=value
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

AIX

HP-UX

Linux

Solaris

```
./imcl install com.ibm.websphere.liberty.v85_offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory installation_directory
-sharedResourcesDirectory shared_directory
-preferences preference_key=value
-properties property_key=value
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

### Tips:

- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `offering_name/lafiles` subdirectory of the installation image or repository for this offering.
- The first time that you install a package by using the Installation Manager, you can specify the shared resources directory. The shared resources directory is where installation artifacts are located that can be used by one or more package groups. It is also used as a staging area for the product payload during installation operations. By default, this content is cached so that it can be used for rollback. Use your largest drive for this installation. You cannot change the directory location until after you uninstall all packages.
- The `offering_version`, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If `offering_version` is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If `offering_version` is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the `-installFixes` argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the `-installFixes` option defaults to all.
  - If the offering version is specified, the `-installFixes` option defaults to none.
- You can add a list of features that are separated by commas:
  - Embeddable EJB container and JPA client (`embeddablecontainer`)  
This option installs the embeddable EJB container and JPA client.  
The embeddable EJB container is a Java Archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans by using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

### Notes:

- If no features are specified, the default feature (`embeddablecontainer`) is installed.



8.5.5.4

Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets

- Liberty features
- To ensure that your installation process completes for systems that do not have internet access specify the `-properties user.feature=""` parameter on the Installation Manager command line. If you do not specify this parameter, the installation process attempts to access the internet and an error occurs.  
ERROR: Unable to connect to the IBM WebSphere Liberty repository or local Installation Manager repository.

Verify that firewalls are configured to allow the Installation Manager to access the internet, or that the local Installation Manager repository can be accessed. If the problem persists, then the repository server might be unavailable. To continue the installation without additional assets, specify the `user.feature=""` and `user.addon=""` parameters.

- **8.5.5.5** You can receive a `NullPointerException` when you apply fix pack V8.5.5.5 for WebSphere Application Server Liberty. The error can occur when you download the WebSphere Application Server Liberty fix pack compressed files and use them, at the downloaded directory location, to install or update Liberty. You can work around this Version 8.5.5.5 error condition by taking the following steps:

- Extract the WebSphere Application Server Liberty fix pack compressed file to a temporary directory.
- Use the temporary directory as the Installation Manager repositories to install or update Liberty.

```
unzip 8.5.5-WS-LIBERTYPROFILE-0S390-FP0000005.zip to /tmp/fp8555
./imcl install
com.ibm.websphere.liberty.v85_8.5.5005.20150305_2214
-installationDirectory /SERVICE/usr/lpp/zWebSphere/Liberty/V8R5
-repositories
/tmp/fp8555
-acceptLicense
```

- **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net. If you want to install additional assets, specify the following properties on the command line. You can specify the short name or symbolic name. Note that the feature short names, such as *FeatureA*, are separated by double commas in the following example:

```
-properties user.feature=FeatureA,,FeatureB,,FeatureC,user.accept.license=true
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

- **8.5.5.4** The following example installs the Extended Programming Models on the `user.addon` parameter, and the Portlet Container feature on the `user.feature` parameter.

```
imcl install com.ibm.websphere.liberty.ND.v85
-properties user.accept.license=true,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory D:\IBM\Liberty -acceptLicense
-repositories D:\IBM\LibertyRepo
-sharedResourcesDirectory D:\IBM\IMShared
-showProgress
```

- **8.5.5.6** You can also install assets from instances of the Liberty Asset Repository Service or local directory-based repositories. For more information about these asset repositories, see “Installing assets using Installation Manager” on page 872. Add the repository on the `-repositories` parameter. The repositories are accessed in the order that they are specified. By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, on the `-properties` parameter, set the `user.useLibertyRepository` option to `false`. **8.5.5.8**

```
imcl install com.ibm.websphere.liberty.v85
-properties user.useLibertyRepository=false,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory D:\IBM\Liberty -acceptLicense
-repositories D:\IBM\LibertyProductRepo,https://your_onprem_asset_repo_url,D:\IBM\LocalAssetRepo,D:\IBM\LocalAssetRepo2.zip
-sharedResourcesDirectory D:\IBM\IMShared
-showProgress
```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- Installation Manager can save earlier versions of a package to roll back to if you experience issues later. When Installation Manager rolls back a package to a previous version, the current versions of the files are uninstalled and the earlier versions are reinstalled. If you choose not to save the files for rollback, you can prevent the files from being saved by using the following preference in your command specification:

```
-preference com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts=False
```

For more information on setting your Installation Manager preferences, see IBM Installation Manager documentation.

**Tip:** Even if you choose not to preserve files locally for rollback, you can still roll back to any previously installed level by accessing the appropriate offering repository.

- You can use the `cic.selector.arch` property key and related value to specify the architecture to install, 32 bit or 64 bit.

Here is an example of specifying a 32-bit architecture:

```
-properties cic.selector.arch=x86
```

If you do not specify anything for this key, you get a correct match for your system. For a 64-bit system, the installation defaults to a 64-bit installation.

Your choice here applies to all packages that are installed in the package group. For information about the supported values for the `cic.selector.key` keys, see the Values for `cic.selector.key` table in the Installation Manager documentation.

- The program might write important post-installation instructions to standard output.

For more information on using the `imcl` command to install the offering, see the IBM Installation Manager documentation.

## Installing Liberty on distributed operating systems by using response files

You can install WebSphere Application Server Liberty Core by using Installation Manager response files.

### Before you begin

Prepare each of the systems onto which you want to install the offering as described in “Installing Installation Manager and preparing to install Liberty” on page 694.

### About this task

Using Installation Manager, you can work with response files to install the offering in various ways. You can record a response file by using the GUI as described in the following procedure, create a new response file, or copy and modify an existing response file.

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

1. Optional: Record a response file to install the offering.

On one of your systems, perform the following actions to record a response file that can install the offering.

- a. From a command line, change to the `eclipse` subdirectory in the directory where you installed Installation Manager.
- b. Start Installation Manager from the command line by using the **-record** option.

For example:

- **Windows** Administrator or non-administrator:

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\install_response_file.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** Administrator:

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/install_response_file.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** Non-administrator:

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/install_response_file.xml
```

**Tip:** When you record a new response file, you can specify the **-skipInstall** parameter. Using this parameter has the following benefits:

- No files are installed, which speeds up the recording.
- If you use a temporary data location with the **-skipInstall** parameter, Installation Manager writes the installation registry to the specified data location while recording. When you start Installation Manager again without the **-skipInstall** parameter, you then can use your response file to install against the real installation registry.

Do not use the **-skipInstall** operation on the agent data location that is used by Installation Manager. This operation is unsupported. Use a clean writable location, and reuse that location for future recording sessions.

For more information, read the IBM Installation Manager documentation.

- c. Follow the instructions that are described in “Installing Liberty on distributed operating systems using the GUI” on page 697.
2. Optional: Create a credential-storage file for installation. If you are using an authenticated remote repository, you can store credentials for URLs that require authentication, such as remote repositories, in a credential-storage file. For IBM Installation Manager Version 1.6.2 and later, use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. For previous versions of Installation Manager, the **-keyring** and **-password** options are used to access credentials in a keyring file. These options are deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures.
  - For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation.
  - For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.
3. Use the response files to install the offering.

Go to a command line on each of the systems on which you want to install the offering, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the offering.

For example:

- **Windows** Administrator or non-administrator:

```
imcl.exe -acceptLicense
input C:\temp\install_response_file.xml
-log C:\temp\install_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

- **AIX** **HP-UX** **Linux** **Solaris** Administrator:

```
./imcl -acceptLicense
input /var/temp/install_response_file.xml
-log /var/temp/install_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

- **AIX** **HP-UX** **Linux** **Solaris** Non-administrator:

```
./imcl -acceptLicense
input user_home/var/temp/install_response_file.xml
-log user_home/var/temp/install_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

## Notes:



- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `offering_name/lafiles` subdirectory of the installation image or repository for this offering.
  - The program might write important post-installation instructions to standard output.
- Read the IBM Installation Manager documentation for more information.

## Example

**Windows** The following is an example of a response file for installing the offering.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean="true" temporary="true">
<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85" />
</server>
<install modify='false'>
<offering id='com.ibm.websphere.liberty.v85'
 profile='WebSphere Liberty V8.5'
 features='embeddablecontainer' installFixes='none' />
</install>
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

**Important:** **AIX** **Linux** **Solaris** **Windows** If you are installing on a 64-bit system, you must include one of the options for an IBM Software Development Kit.

- You can use the `cic.selector.arch` property key and related value to specify the architecture to install, 32 bit or 64 bit.

Here is an example of specifying a 32-bit architecture:

```
<profile>
...
<data key='cic.selector.arch' value='x86' />
...
</profile>
```

If you do not specify anything for this key, you receive a correct match for your system. For a 64-bit system, the installation defaults to a 64-bit installation.

Your choice here applies to all packages that are installed in the package group. For information about the supported values for the `cic.selector.key` keys, see the Values for `cic.selector.key` table in the Installation Manager documentation.

To disable remote searches for updates in the response file, set the following preferences to false:

- **offering.service.repositories.areUsed**  
Used for searching remote repositories for updates to installed offerings
- **com.ibm.cic.common.core.preferences.searchForUpdates**  
Used for searching for updates to Installation Manager

For example:

```
<preference value='false' name='offering.service.repositories.areUsed' />
<preference value='false' name='com.ibm.cic.common.core.preferences.searchForUpdates' />
```

You can find more details on silent preference keys in the IBM Installation Manager documentation.

The following examples show you how to change response file in order to perform alternative actions.

- To install multiple copies of this offering, specify a different installation location and a new package group for each installation. For example, to install a second copy of the offering into the `C:\Program Files\IBM\WebSphere\Liberty_2` directory and create the `WebSphere Liberty V8.5_2` package group, replace:

```
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty_2'>
```

with:

```
<profile id='WebSphere Liberty V8.5.2'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty_2'>
```

- To add optional features, add them as an entry in a comma-separated list. For example:

```
<offering id='com.ibm.websphere.liberty.v85'
 profile='WebSphere Liberty V8.5'
 features='embeddablecontainer' installFixes='none' />
```


- Embeddable EJB container and JPA client (`embeddablecontainer`)

This option installs the embeddable EJB container and JPA client.

The embeddable EJB container is a Java archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans by using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

If no features are specified, the default feature (`embeddablecontainer`) is installed.

-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install Liberty Repository assets, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later. Previous versions of Installation Manager do not have the option to install Liberty Repository assets. If you use a response file and did not update Installation Manager to Version 1.6.2 or later, the assets that you specify in the response file are ignored during installation.

If you want to install additional features, specify two extra data key elements in your response file. You can use either the symbolic name or the short name.


The following example installs the Portlet Container and Portlet Serving features using the symbolic name.

```
<data key='user.feature' value='com.ibm.websphere.appserver.portlet-2.0,,com.ibm.websphere.appserver.portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

The following example installs the Portlet Container and Portlet Serving features using the short name:

```
<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.


-  **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features

The following example installs the Extended Programming Models using the `user.addon` parameter and the Portlet Container and Portlet Serving features using the `user.feature` parameter with short names:

```
<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

-  **8.5.5.6** You can also install assets from an instance of the Liberty Asset Repository Service or local directory-based repositories. For more information, see “Installing assets using Installation Manager” on page 872. Add the repository on repository elements. If Installation Manager does not recognize the repository, point directly to the `repository.config` file. When you install assets, the repositories are accessed in the order that you specify them.

```

<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85" />
<repository location="https://your_onprem_asset_repo_url" />
<repository location="D:\IBM\LocalAssetRepo" />

```

8.5.5.8

<repository location="D:\IBM\LocalAssetRepo2.zip" /> </server>By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, set the **user.useLibertyRepository** parameter to false:

```

<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.useLibertyRepository' value='false' />

```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- Installation Manager can save earlier versions of a package to roll back to if you experience issues later. When Installation Manager rolls back a package to a previous version, the current versions of the files are uninstalled and the earlier versions are reinstalled. If you choose not to save the files for rollback, you can prevent the files from being saved by changing the following preference in your response file:

```
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
```

to this:

```
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='false' />
```

For more information on setting your Installation Manager preferences, see the IBM Installation Manager documentation.

**Tip:** Even if you choose not to preserve files locally for rollback with this option, you can still roll back to any previously installed level by accessing the appropriate product repository.

## Adding and removing features from Liberty on distributed operating systems

You can use Installation Manager to install and remove WebSphere Application Server Liberty Core features.

### Before you begin

Ensure that your Installation Manager preferences are pointing to the appropriate web-based or local repositories containing WebSphere Application Server Liberty Core.

### About this task

You can use the Installation Manager to install or remove features using one of the following procedures:

- Using the GUI
- Using a silent response file

You can record this response file using the GUI and Installation Manager's record mode, or you can manually create or modify a response file to suit your needs.

- Using the **imcl** command-line tool

Go to the IBM Installation Manager Information Center for more information.

### Procedure

1. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being modified.
2. Start Installation Manager.
3. Click **Modify**.
4. Select the package group to modify.
5. Click **Next**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

- Expand IBM WebSphere Application Server LibertyCore.
- Select the appropriate checkbox to install a feature, or clear the appropriate checkbox to remove a feature if you already have it installed.

- Embeddable EJB container and JPA client

This option installs the embeddable EJB container and JPA client.

The embeddable EJB container is a Java archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

- Click **Next**.
- Review the summary information, and click **Modify**.
  - If the modification is successful, the program displays a message indicating that installation is successful.
  - If the modification is not successful, click **View Log File** to troubleshoot the problem.

10. Click **Finish**.

11. Click **File > Exit** to close Installation Manager.

## Examples

In the following examples, the optional feature offering names are enclosed in parentheses; for example: Embeddable EJB container and JPA client (embeddablecontainer)

- **Windows** Example of a response file that adds a feature to an installation:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>
<server>
<repository location='http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85'>
</repository>
<install modify='true'>
<offering id='com.ibm.websphere.liberty.v85'
 profile='WebSphere Liberty V8.5'
 features='embeddablecontainer'>
</install>
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

- **Windows** Example of the **imcl** command that is modified to add features to an installation:

```
imcl.exe modify com.ibm.websphere.liberty.v85
-addFeatures embeddablecontainer
-repositories http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-installationDirectory C:\Program Files\IBM\WebSphere\Liberty
-secureStorageFile C:\credential.store -masterPasswordFile C:\master_password_file.txt
```

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

- Example of the **imcl** command that is modified to remove a feature from an installation:

```
imcl.exe modify com.ibm.websphere.liberty.v85
-removeFeatures embeddablecontainer
-repositories http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-installationDirectory C:\Program Files\IBM\WebSphere\Liberty
-secureStorageFile C:\credential.store -masterPasswordFile C:\master_password_file.txt
```

## Uninstalling Liberty from distributed operating systems using the GUI

You can use the Installation Manager GUI to uninstall WebSphere Application Server Liberty Core.

### Procedure

1. Uninstall WebSphere Application Server Liberty Core.
  - a. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
  - b. Start Installation Manager.
  - c. Click **Uninstall**.
  - d. In the **Uninstall Packages** window, perform the following actions.
    - 1) Select **IBM WebSphere Application Server Liberty Core** and the appropriate version.

**Note:** If you are uninstalling the trial version of this offering, select **IBM WebSphere Application Server Liberty Core Trial** and the appropriate version.

When you uninstall the IBM WebSphere Application Server Liberty Core package, you must uninstall all the packages under the same package group that are extensions to the IBM WebSphere Application Server Liberty Core package. IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty is such an extension.

- 2) Click **Next**.
  - e. Review the summary information.
  - f. Click **Uninstall**.
    - If the uninstallation is successful, the program displays a message that indicates success.
    - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
  - g. Click **Finish**.
  - h. Click **File > Exit** to close Installation Manager.
2. Optional: Uninstall IBM Installation Manager.

**Important:** Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

Read the IBM Installation Manager Information Center for information about performing this procedure.

## Uninstalling Liberty from distributed operating systems using the command line

You can uninstall WebSphere Application Server Liberty Core using the Installation Manager command line.

### Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
4. Use the `imcl` command to uninstall the offering.

Windows

```
imcl.exe uninstall com.ibm.websphere.liberty.v85 -installationDirectory installation_directory
```

AIX

HP-UX

Linux

Solaris

```
./imcl uninstall com.ibm.websphere.liberty.v85 -installationDirectory installation_directory
```

### Tips:

- Use the **imcl modify** command to add or remove features.
- When you uninstall the IBM WebSphere Application Server Liberty Core package, you must uninstall all the packages under the same package group that are extensions to the IBM WebSphere Application Server Liberty Core package. IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty is such an extension.

Go to the IBM Installation Manager Information Center for more information.

## 5. Optional: Uninstall IBM Installation Manager.

**Important:** Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

Read the IBM Installation Manager Information Center for information about using the uninstall script to perform this procedure.

## Uninstalling Liberty from distributed operating systems by using response files

You can uninstall WebSphere Application Server Liberty Core by using Installation Manager response files.

### Before you begin

**Optional:** Perform or record the installation of Installation Manager and installation of the product to a temporary installation registry on one of your systems so that you can use this temporary registry to record the uninstall without using the standard registry where Installation Manager is installed.

### About this task

Using Installation Manager, you can work with response files to uninstall the product in various ways. You can record a response file by using the GUI as described in the following procedure. Or, you can generate a new response file by hand or by taking an example and modifying it.

### Procedure

1. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
2. **Optional: Record a response file to uninstall the product:** On one of your systems, perform the following actions to record a response file that uninstalls the product:
  - a. From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.
  - b. Start Installation Manager from the command line by using the **-record** option.

For example:

- **Windows Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\uninstall_response_file.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/uninstall_response_file.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/uninstall_response_file.xml
```

**Tip:** If you choose to use the **-skipInstall** parameter with a temporary installation registry created as described in *Before you begin*, Installation Manager uses the temporary installation registry while recording the response file. It is important to note that when the **-skipInstall** parameter is specified, no product packages are installed or uninstalled. All of the actions that you perform in Installation Manager update the installation data that is stored in the specified

temporary registry. After the response file is generated, it can be used to uninstall the product, removing the product files and updating the standard installation registry.

Do not use the **-skipInstall** operation on the actual agent data location that is used by Installation Manager. This operation is unsupported. Use a clean writable location, and reuse that location for future recording sessions.

For more information, read the IBM Installation Manager Information Center.

- c. Click **Uninstall**.
- d. In the **Uninstall Packages** window, perform the following actions.
  - 1) Select **IBM WebSphere Application Server Liberty Core** and the appropriate version.

**Note:** If you are uninstalling the trial version of this offering, select **IBM WebSphere Application Server Liberty Core Trial** and the appropriate version.

- 2) Click **Next**.
- e. Review the summary information.
- f. Click **Uninstall**.
  - If the uninstallation is successful, the program displays a message that indicates success.
  - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
- g. Click **Finish**.
- h. Click **File > Exit** to close Installation Manager.

3. **Use the response file to uninstall the product:** From a command line on each of the systems from which you want to uninstall the product, change to the `eclipse/tools` sub-directory in the directory where you installed Installation Manager and use the response file that you created to uninstall the product.

For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\uninstall_response_file.xml
-log C:\temp\uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/uninstall_response_file.xml
-log /var/temp/uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/uninstall_response_file.xml
-log user_home/var/temp/uninstall_log.xml
```

Go to the IBM Installation Manager Information Center for more information.

4. **Optional: List all installed packages to verify the uninstallation.**

**AIX HP-UX Linux Solaris**

```
./imcl listInstalledPackages
```

**Windows**

```
imcl listInstalledPackages
```

5. **Optional: Uninstall IBM Installation Manager.**

**Important:** Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

Read the IBM Installation Manager Information Center for information about using the uninstall script to perform this procedure.

**Windows**

## Example

The following is an example of a response file for uninstalling the product.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>
<uninstall modify='false'>
<offering id='com.ibm.websphere.liberty.v85'
 profile='WebSphere Liberty V8.5' />
</uninstall>
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

**Tip:** When you uninstall the IBM WebSphere Application Server LibertyCore package, you must uninstall all the packages under the same package group that are extensions to the IBM WebSphere Application Server LibertyCore package. IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty is such an extension. To uninstall the entire package group, remove the id attribute in the <offering ../> stanza (id='com.ibm.websphere.liberty.v85' in the previous example) from the response file. Installation Manager then uninstalls all of the packages under the package group.

## Installing and uninstalling Liberty interim fixes and fix packs on distributed operating systems

Interim fixes provide corrective service for specific known problems. Fix packs contain bundled service to bring WebSphere Application Server Liberty Core up to a new level. You can use IBM Installation Manager to update the offering with the interim fixes and fix packs that are available for your service level of WebSphere Application Server Liberty Core.

### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty Core. The most current information is available from the IBM Software Support Center and Fix Central.

IBM Installation Manager is used to apply and remove maintenance.

**Important:** Installation Manager can install any fix-pack level of the product directly without installing the intermediate fix packs; in fact, Installation Manager installs the latest level by default. For example, you can skip fix-pack levels and go from Version 8.5.5.1 directly to Version 8.5.5.5. Keep in mind, however, that later you can not roll back to any level that was skipped. If you directly install to Version 8.5.5.5, for example, you cannot roll back to Version 8.5.5.4. If you skip from Version 8.5.5.1 to Version 8.5.5.5, you can only roll back to Version 8.5.5.1. You should plan your installations accordingly.

- | **New:** Fix pack 16.0.0.2 is the next fix pack after 8.5.5.9. You can continue to use the same Version 8.5
- | Installation Manager repositories and offering IDs to install or update to 16.0.0.2, or you can use the new
- | versionless repositories and offerings. For more information about fix pack 16.0.0.2, see What is new in
- | Liberty in the new location of the latest Liberty documentation.





### About this task

**Tip:** You can use the IBM Packaging Utility to generate a new local or web-based repository that contains all of the fixes that you want to install and then use Installation Manager to update WebSphere Application Server Liberty Core with all of the interim fixes as a group. For information on using the Packaging Utility, see the IBM Installation Manager Information Center.

### Procedure

- “Installing Liberty interim fixes on distributed operating systems using the GUI” on page 713
- “Installing Liberty interim fixes on distributed operating systems using the command line” on page 715
- “Uninstalling Liberty interim fixes from distributed operating systems using the GUI” on page 717



- “Uninstalling Liberty interim fixes from distributed operating systems using the command line” on page 718
- “Installing Liberty fix packs on distributed operating systems using the GUI” on page 718
- “Installing Liberty fix packs on distributed operating systems using the command line” on page 720
- “Installing Liberty fix packs on distributed operating systems using response files” on page 724
- “Uninstalling Liberty fix packs from distributed operating systems using the GUI” on page 729
- “Uninstalling Liberty fix packs from distributed operating systems using the command line” on page 730
- “Uninstalling Liberty fix packs from distributed operating systems by using response files” on page 731
-     Optional: After the fix pack is installed or removed, reapply any ownership and permissions changes that were made after the offering was originally installed.

### Installing Liberty interim fixes on distributed operating systems using the GUI:

You can use the IBM Installation Manager graphical user interface (GUI) to update the offering with the interim fixes that are available for your service level of WebSphere Application Server Liberty Core.

#### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty Core. The most current information is available from the IBM Software Support Center and Fix Central.

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the IBM Installation Manager Information Center.

#### Procedure

1. For a list of fixes that are available for WebSphere Application Server Liberty Core and specific information about each fix, perform the following actions.
  - a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server Liberty Core** as the product.
  - d. Select the installed version.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. **Recommendation:** Make a list of the names of the fixes that you would like to install.
2. Update WebSphere Application Server Liberty Core with the fixes using one of the following procedures.
  - Access the live service repository that contains the fixes, and use web-based updating. Use Installation Manager on your local system to update WebSphere Application Server Liberty Core with the interim fixes from the live web-based service repositories.
    - For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are listed in Online product repositories for WebSphere Application Server offerings.
    - These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the offering.

To install a fix from a service repository, perform the following actions:

- a. Log on to your system.
- b. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- c. Start Installation Manager.
- d. Click **Update**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.

- e. Select the package group to update with the fix.

**Tip:** If the **Search service repositories during installation and updates** option is selected on the Installation Manager Repository preference page and you are connected to the Internet, you can click **Check for Other Versions and Extensions** to search for updates in the default update repositories for the selected packages. In this case, you do not need to add the specific service-repository URL to the Installation Manager Repository preference page.

- f. Click **Next**.

- g. Select the fixes to install, and click **Next**.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

- h. Review the summary information, and click **Update**.

- i. Click **Finish**.

- j. Click **File > Exit** to close Installation Manager.

- Download the files that contain the fixes from Fix Central, and use local updating.

You can download compressed files that contain the fixes from Fix Central. Each compressed fix file contains an Installation Manager repository for the fix and usually has a .zip extension. After downloading the fix files, you can use one of the following procedures to update WebSphere Application Server Liberty Core:

- Use Installation Manager to update WebSphere Application Server Liberty Core with the fixes.
- Use the IBM Packaging Utility to generate a new repository that contains all of the fix files that you downloaded, then use Installation Manager to update WebSphere Application Server Liberty Core with all of the fixes as a group. For information on using the Packaging Utility, see the IBM Installation Manager Information Center.

- a. To download the fixes, perform the following actions:

- 1) Go to Fix Central.
- 2) Select **WebSphere** as the product group.
- 3) Select **WebSphere Application Server Liberty Core** as the product.
- 4) Select the installed version.
- 5) Select your operating system as the platform, and click **Continue**.
- 6) Select **Browse for fixes**, and click **Continue**.
- 7) Select the fixes that you want to download, and click **Continue**.
- 8) Select your download options, and click **Continue**.
- 9) Click **I agree** to agree to the terms and conditions.
- 10) Click **Download now** to download the fixes.

- b. To install a fix from a downloaded compressed file, perform the following actions:

- 1) Log on to your system.

- 2) Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- 3) Start Installation Manager.
- 4) Add the location of the fix file that you downloaded to your Installation Manager preferences.
- 5) Click **Update**.
- 6) Select the package group to update with the fix, and click **Next**.
- 7) Select the fixes to install, and click **Next**.  
Any recommended fixes are selected by default.  
If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.
- 8) Review the summary information, and click **Update**.
- 9) Click **Finish**.
- 10) Click **File > Exit** to close Installation Manager.

### Installing Liberty interim fixes on distributed operating systems using the command line:

You can use the IBM Installation Manager command line to update the offering with the interim fixes that are available for your service level of WebSphere Application Server LibertyCore.

#### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty Core. The most current information is available from the IBM Software Support Center and Fix Central.

IBM Installation Manager is used to apply maintenance to WebSphere Application Server LibertyCore.

#### Procedure

1. For a list of interim fixes and fix packs that are available for WebSphere Application Server Liberty Core and specific information about each fix, perform the following actions.
  - a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server LibertyCore** as the product.
  - d. Select the version of the offering to be updated.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. **Recommendation:** Make a list of the names of the fixes that you would like to install.
2. Update WebSphere Application Server LibertyCore with the interim fixes using one of the following procedures.
  - **Access the live service repository that contains the interim fixes, and use web-based updating.**  
Use Installation Manager on your local system to update WebSphere Application Server Liberty Core with the interim fixes from the live web-based service repositories.
    - For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are listed in Online product repositories for WebSphere Application Server offerings.
    - These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the offering.

To install an interim fix from a service repository, perform the following actions:

- a. Log on to your system.
- b. If you do not already have Installation Manager credential-storage and master-password files containing your IBM software user ID and password, create files that will allow you to access the repository.

**Note:** These are the credentials that you use to access protected IBM software websites. For information on creating credential-storage and master-password files for Installation Manager, read the IBM Installation Manager Information Center.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

- c. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- d. Change to the `Installation_Manager_binaries/eclipse/tools` directory, where `Installation_Manager_binaries` is the installation root directory for the Installation Manager.
- e. Install the interim fix.

```
AIX HP-UX Linux Solaris

./imcl install fix_name
 -installationDirectory offering_installation_location
 -repositories repository_URL
 -secureStorageFile storage_file -masterPasswordFile master_password_file
```

```
Windows

imcl.exe install fix_name
 -installationDirectory offering_installation_location
 -repositories repository_URL
 -secureStorageFile storage_file -masterPasswordFile master_password_file
```

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

- f. **Optional:** List all installed packages to verify the installation:

```
AIX HP-UX Linux Solaris

./imcl listInstalledPackages -long

Windows

imcl.exe listInstalledPackages -long
```

- **Download the files that contain the interim fixes from Fix Central, and use local updating.**

You can download compressed files that contain the interim fixes from Fix Central. Each compressed fix file contains an Installation Manager repository for the interim fix and usually has a `.zip` extension. After downloading the fix files, you can use one of the following procedures to update WebSphere Application Server Liberty Core:

- Use Installation Manager to update WebSphere Application Server Liberty Core with the interim fixes.
- Use the IBM Packaging Utility to generate a new repository that contains all of the fix files that you downloaded, then use Installation Manager to update WebSphere Application Server

LibertyCore with all of the interim fixes as a group. For information on using the Packaging Utility, see the IBM Installation Manager Information Center.

- a. To download the interim fixes, perform the following actions:
  - 1) Go to Fix Central.
  - 2) Select **WebSphere** as the product group.
  - 3) Select **WebSphere Application Server LibertyCore** as the product.
  - 4) Select the version of the offering to be updated.
  - 5) Select your operating system as the platform, and click **Continue**.
  - 6) Select **Browse for fixes**, and click **Continue**.
  - 7) Select the interim fixes that you want to download, and click **Continue**.
  - 8) Select your download options, and click **Continue**.
  - 9) Click **I agree** to agree to the terms and conditions.
  - 10) Click **Download now** to download the interim fixes.
- b. To install an interim fix from a downloaded compressed file, perform the following actions:
  - 1) Log on to your system.
  - 2) Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
  - 3) Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.
  - 4) Install the interim fix.

AIX

HP-UX

Linux

Solaris

```
./imcl install fix_name
-installationDirectory offering_installation_location
-repositories compressed_file
```

Windows

```
imcl.exe install fix_name
-installationDirectory offering_installation_location
-repositories compressed_file
```

- 5) **Optional:** List all installed packages to verify the installation:

AIX

HP-UX

Linux

Solaris

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

### Uninstalling Liberty interim fixes from distributed operating systems using the GUI:

You can use the IBM Installation Manager graphical user interface (GUI) to remove interim fixes from WebSphere Application Server Liberty Core.

#### Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
3. Start Installation Manager.
4. Click **Uninstall**.
5. Select the interim fixes to uninstall.
6. Click **Next**.

7. Review the summary information, and click **Uninstall**.
  - If the uninstallation is successful, the program displays a message indicating that the uninstallation is successful.
  - If the uninstallation is not successful, click **View Log File** to troubleshoot the problem.
8. Click **Finish**.
9. Click **File > Exit** to close Installation Manager.

### Uninstalling Liberty interim fixes from distributed operating systems using the command line:

You can use the IBM Installation Manager command-line function to remove interim fixes from WebSphere Application Server Liberty Core.

#### Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
3. Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.
4. Uninstall the interim fix:

AIX

HP-UX

Linux

Solaris

```
./imcl uninstall interim_fix_name
 -installationDirectory offering_installation_location
```

Windows

```
imcl.exe uninstall interim_fix_name
 -installationDirectory offering_installation_location
```

### Installing Liberty fix packs on distributed operating systems using the GUI:

You can update WebSphere Application Server Liberty Core to a later version using the Installation Manager GUI.

#### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty Core. The most current information is available from the IBM Software Support Center and Fix Central.

8.5.5.4

To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the IBM Installation Manager Information Center.



8.5.5.6

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

#### Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.

3. Start Installation Manager.
4. Click **Update**.

**Note:** If prompted to authenticate, specify the IBM ID and password that you use to access protected IBM software websites.


5. Select the package group to update. If you select **Update all**, Installation Manager searches all of the added and predefined repositories for recommended updates to all of the package groups that it has installed.
  - Use this feature only if you have full control over which fixes are contained in the targeted repositories.
    - If you create and point to a set of custom repositories that include only the specific recommended fixes that you want to install, you should be able to use this feature confidently.
    - If you enable searching service repositories or install fixes directly from other live web-based repositories, you might not want to select this option so that you can select only the fixes that you want to install for each offering on subsequent panels.
  - If you select **Update all**, Installation Manager will install only the recommended updates to all of the package groups; it will not allow you to select non-recommended fixes for installation. If you want to install non-recommended fixes, perform the following actions:
    - a. On this panel, clear the **Update all** check box and select an offering to update.
    - b. On the next panel, clear the option to show only recommended fixes and then select the fixes that you want to install.
  - **8.5.5.4** The **Update all** option might generate a warning message if you complete the following steps:
    - a. Install WebSphere Application Server Liberty v8.5.5.3 with the Extended Programming Model.
    - b. Install WebSphere Application Server Network Deployment Liberty v8.5.5.3 with the Extended Programming Model.
    - c. Select **Update all**. Click **Next**.
    - d. You might receive an Update Validation Warning message with multiple messages. Click **OK**. Click **Next**.
    - e. You might receive the Extended Programming Model Warning message panel.
    - f. Click **Back** to the Update Package panel. Click **Next**.
    - g. You might again receive an Update Validation Warning message with multiple messages.
    - h. From the Extended Programming Model Warning message panel, click **Next**. In the Offering License panel, find the following error:

An error has occurred. See error log for more details. Bundle "reference:file:/C:/ProgramData/IBM/InstallationTo fix this problem, complete the following steps:
    - a. Click **Cancel**.
    - b. Select **Update all**.
    - c. Do not click **Back**. This action causes the error that uninstalls the license.
6. Click **Next**.
7. Select the version to which you want to update under **IBM WebSphere Application Server Liberty Core**.
8. Select any fixes that you want to install. Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.
9. Click **Next**.
10. Accept the terms in the license agreements, and click **Next**.
11. Select the optional features that you want in your updated installation.

**8.5.5.4**

Beginning with Version 8.5.5.4, the Extended Programming Model is available from the Liberty Repository. This asset provides a rich set of programming models such as Web Services, JMS (including Message-Driven Beans), and MongoDB 2.0. To install this asset, use the Asset Selection Wizard as described in the next step.

12.  **8.5.5.2** Optional: Install additional Liberty Repository assets. If you do not want to select any additional assets, you can skip this step. To install Liberty Repository assets, you must have IBM Installation Manager Version 1.6.2 or later. To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.
  - a. Select whether you want to install assets from the Liberty Repository, and click **Next**. To install assets from the IBM WebSphere Liberty Repository, you must have access to the internet.
 

**8.5.5.6** If you choose not to connect to the IBM WebSphere Liberty Repository, you can still install assets from configured directory-based repositories or an instance of the Liberty Asset Repository Service. For more information, see “Installing assets using Installation Manager” on page 872.
  - b. **8.5.5.4** Click **Launch Asset Selection Wizard**.
  - c. You can perform a case-insensitive search for assets by display name or description. If you search with the **Keyword** field empty, then the search displays all applicable assets.
 

**8.5.5.5** With Version 8.5.5.5 and later, you can also search for assets by short name.
  - d. Select each asset that you want to install, then click **Next**.
  - e. Accept the license agreement and click **Finish**.
13. Review the summary information, and click **Update**.
  - If the installation is successful, the program displays a message indicating that installation is successful.
  - If the installation is not successful, click **View Log File** to troubleshoot the problem.
14. Click **Finish**.
15. Click **File > Exit** to close Installation Manager.

### Installing Liberty fix packs on distributed operating systems using the command line:

You can use the IBM Installation Manager command line to update the product with the fix packs that are available for WebSphere Application Server Liberty Core.

#### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty Core. The most current information is available from the IBM Software Support Center and Fix Central.

**8.5.5.6**

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

**Tip:** As an alternative to the procedure that is described in this article, Installation Manager allows you to use the **updateAll** command in a response file or on the command line to search for and update all installed packages. Use this command only if you have full control over which fixes are contained in the targeted repositories. You can create and point to a set of custom repositories that include only the specific fixes that you want to install. If you enable searching service repositories or install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install using the **-installFixes** option with the **install** command on the command line or the **installFixes** attribute in a response file.



## About this task

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

1. For a list of interim fixes and fix packs that are available for WebSphere Application Server Liberty Core and specific information about each fix, perform the following actions.
  - a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server** as the product.
  - d. Select the installed version.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. **Recommendation:** Make a note of the name of the fix pack that you would like to install.
2. Update WebSphere Application Server LibertyCore with the fix pack using one of the following procedures.
  - Access the live service repository that contains the fix pack, and use web-based updating. Use Installation Manager on your local system to update WebSphere Application Server Liberty Core with the interim fixes from the live web-based service repositories.
    - For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are based on the following pattern:  
`http://www.ibm.com/software/repositorymanager/offering_ID`

where *offering\_ID* is the offering ID that you can find in WebSphere Application Server product offerings for supported operating systems.

- These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the product.

To install a fix pack from a service repository, perform the following actions:

- a. Log on to your system.
- b. Create Installation Manager credential-storage and master-password files that contain your IBM software user ID and password. These files enable you to access the repository and protected IBM software websites.

When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** For Installation Manager Version 1.6.2 and later, use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In previous versions, the `-keyring` and `-password` options are used to access credentials in a keyring file. These options are deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures.

- For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation.
  - For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.
- c. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.

- d. Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.
- e. Install the fix pack.

**AIX** **HP-UX** **Linux** **Solaris**

```
./imcl install offering_ID offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory offering_installation_location
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

**Windows**

```
imcl.exe install offering_ID offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory offering_installation_location
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```


where *offering\_ID* is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.

#### Tips:

- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
imcl listAvailablePackages -repositories source_repository
```


- You can also specify none, recommended or all with the `-installFixes` argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the `-installFixes` option defaults to all.
  - If the offering version is specified, the `-installFixes` option defaults to none.
- You can add a list of features that are separated by commas.
-  **8.5.5.2** You can specify additional features to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

**Note:** To install Liberty Repository assets, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later.

If you want to install additional features, specify the following properties in the command line. Note that the feature short names, such as *FeatureA*, are separated by double commas:

```
-properties user.feature=FeatureA,,FeatureB,,FeatureC,user.accept.license=true
```

For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

-  **8.5.5.4** Beginning with Version 8.5.5.4, the Extended Programming Models are available to download from the Liberty Repository as the `extendedPackage-1.0` addon. This addon provides a rich set of programming models such as Web Services, JMS (including Message-Driven Beans), and MongoDB 2.0. To install the addon, specify the `extendedPackage-1.0` addon on the **user.addon** option of the **-properties** parameter:

```

imcl install com.ibm.websphere.liberty.ND.v85
-properties user.acceptLicense=true,user.addon=extendedPackage-1.0
-installationDirectory D:\IBM\Liberty -acceptLicense
-repositories D:\IBM\LibertyRepo
-sharedResourcesDirectory D:\IBM\IMShared
-showProgress

```



8.5.5.6

You can also install assets from instances of the Liberty Asset Repository Service or local directory-based repositories. For more information about these asset repositories, see “Installing assets using Installation Manager” on page 872. Add the repository on the **-repositories** parameter. The repositories are accessed in the order that they are specified. By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, on the **-properties** parameter, set the **user.useLibertyRepository** option to false.

8.5.5.8

```

imcl install com.ibm.websphere.liberty.v85
-properties user.useLibertyRepository=false,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory D:\IBM\Liberty -acceptLicense
-repositories D:\IBM\LibertyProductRepo,https://your_onprem_asset_repo_url,D:\IBM\LocalAssetRepo,D:\IBM\LocalAssetRepo2.zip
-sharedResourcesDirectory D:\IBM\IMShared
-showProgress

```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

**Note:** If you do not have the Extended Programming Model installed prior to updating to Version 8.5.5.4, you might encounter the following error during the update:

8.5.5.4

```

java.io.IOException: Too many open files
at java.io.File.createNewFile(File.java:894)

```

To resolve this problem, increase the number of files using the **ulimit** command; for example:

```
ulimit -n 8192
```

- f. List all installed packages to verify the installation:

AIX

HP-UX

Linux

Solaris

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

- Download the file that contains the fix pack from Fix Central, and use local updating. You can download a compressed file that contains the fix pack from Fix Central. Each compressed fix-pack file contains an Installation Manager repository for the fix pack and usually has a .zip extension. After downloading and extracting the fix-pack file, use Installation Manager to update WebSphere Application Server Liberty Core with the fix pack.
  - a. To download the fix pack, perform the following actions:
    - 1) Go to Fix Central.
    - 2) Select **WebSphere** as the product group.
    - 3) Select **WebSphere Application Server** as the product.
    - 4) Select the installed version.
    - 5) Select your operating system as the platform, and click **Continue**.
    - 6) Select **Browse for fixes**, and click **Continue**.
    - 7) Select the fix pack that you want to download, and click **Continue**.
    - 8) Select your download options, and click **Continue**.
    - 9) Click **I agree** to agree to the terms and conditions.
    - 10) Click **Download now** to download the fix pack.
    - 11) Transfer the compressed file in binary format to the system on which it is installed.
    - 12) Extract the compressed repository files to a directory on your system.
  - b. To install a fix pack from a downloaded file, perform the following actions:
    - 1) Log on to your system.

- 2) Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- 3) Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.
- 4) Install the fix pack.

AIX      HP-UX      Linux      Solaris

```
./imcl install offering_ID offering_version,optional_feature_ID
-installationDirectory offering_installation_location
-repositories location_of_expanded_files
-acceptLicense
```

Windows

```
imcl.exe install offering_ID offering_version,optional_feature_ID
-installationDirectory offering_installation_location
-repositories location_of_expanded_files
-acceptLicense
```

#### Tips:

- The *offering\_ID* is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.
- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the *-installFixes* argument to indicate which interim fixes you want installed with the offering.
    - If the offering version is **not** specified, the *-installFixes* option defaults to all.
    - If the offering version is specified, the *-installFixes* option defaults to none.
  - You can add a list of features that are separated by commas.
- 5) **Optional:** List all installed packages to verify the installation:

AIX      HP-UX      Linux      Solaris

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

#### Installing Liberty fix packs on distributed operating systems using response files:

You can update WebSphere Application Server Liberty Core to a later version using Installation Manager response files.

#### About this task

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.



If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

**Tip:** As an alternative to the procedure that is described in this topic, use the Installation Manager **updateAll** command in a response file or on the command line to search for and update all installed packages. Use this command only if you have full control over which fixes are contained in the targeted repositories. Create and point to a set of custom repositories that include only the specific fixes that you want to install. If you enable searching service repositories or if you install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install using the **-installFixes** option with the **install** command on the command line or the **installFixes** attribute in a response file.

### Procedure

1. To obtain a list of interim fixes and fix packs that are available for WebSphere Application Server Liberty Core installation and specific information about each fix, perform the following actions.
  - a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server** as the product.
  - d. Select the installed version.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. Note of the name of the fix pack that you would like to install.
2. Update WebSphere Application Server LibertyCore with the fix pack using one of the following procedures.
  - Access the live service repository that contains the fix pack, and use web-based updating. Use Installation Manager on your local system to update WebSphere Application Server Liberty Core with the interim fixes from the live web-based service repositories.
    - For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are based on the following pattern:  
`http://www.ibm.com/software/repositorymanager/offering_ID`

where *offering\_ID* is the offering ID that you can find in WebSphere Application Server product offerings for supported operating systems.

- These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the product.

Perform the following actions:

- a. Log on to your system.
- b. Create files that enable you to access the repository. Installation Manager credential-storage and master-password files contain your IBM software user ID and password and enable you to access protected IBM software websites. When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the **imutilsc** command is unable to find the URL that is specified.

**Note:** For Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In previous versions of Installation Manager, the **-keyring** and **-password** options are used to access credentials in a keyring file. These options are deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures.

- For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation.
- For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.
- c. Stop all servers and applications on the WebSphere Application Server installation that is being updated.
- d. Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.
- e. Install the fix pack using a response file.

For example:

-  Administrator or non-administrator:

```
imcl.exe -acceptLicense
input C:\temp\update_response_file.xml
-log C:\temp\update_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

-     Administrator:

```
./imcl -acceptLicense
input /var/temp/update_response_file.xml
-log /var/temp/update_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

-     Non-administrator:

```
./imcl -acceptLicense
input user_home/var/temp/update_response_file.xml
-log user_home/var/temp/update_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

- Download the file that contains the fix pack from Fix Central, and use local updating.

You can download a compressed file that contains the fix pack from Fix Central. Each compressed fix-pack file contains an Installation Manager repository for the fix pack and usually has a .zip extension. After downloading and extracting the fix-pack file, use Installation Manager to update WebSphere Application Server Liberty with the fix pack.

- a. To download the fix pack, perform the following actions:

- 1) Go to Fix Central.
- 2) Select **WebSphere** as the product group.
- 3) Select **WebSphere Application Server** as the product.
- 4) Select the installed version.
- 5) Select your operating system as the platform, and click **Continue**.
- 6) Select **Browse for fixes**, and click **Continue**.
- 7) Select the fix pack that you want to download, and click **Continue**.
- 8) Select your download options, and click **Continue**.
- 9) Click **I agree** to agree to the terms and conditions.
- 10) Click **Download now** to download the fix pack.
- 11) Transfer the compressed file in binary format to the system on which it will be installed.
- 12) Extract the compressed repository files to a directory on your system.

- b. Perform the following actions:

- 1) Log on to your system.
- 2) If the repository requires a username and password, create a credential-storage file to access this repository.

For more information on creating a credential-storage file for Installation Manager, read the IBM Installation Manager Information Center.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutils` command is unable to find the URL that is specified.

- 3) Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- 4) Change to the `Installation_Manager_binaries/eclipse/tools` directory, where `Installation_Manager_binaries` is the installation root directory for the Installation Manager.
- 5) Install the fix pack using a response file.

For example:

— **Windows Administrator or non-administrator:**

```
imcl.exe -acceptLicense
input C:\temp\update_response_file.xml
-log C:\temp\update_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

— **AIX HP-UX Linux Solaris Administrator:**

```
./imcl -acceptLicense
input /var/temp/update_response_file.xml
-log /var/temp/update_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

— **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl -acceptLicense
input user_home/var/temp/update_response_file.xml
-log user_home/var/temp/update_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

## Example

**Windows** The following is an example of a response file for updating the product to a later version.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
<repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85'/>
</server>
<profile id='WebSphere Liberty V8.5'
installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
<install modify='false'>
<offering profile='WebSphere Liberty V8.5' id='com.ibm.websphere.liberty.v85'
version='8.5.5.20101025_2108'/>
</install>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared'/>
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
</agent-input>
```


## Tips:

- The profile ID (`<profile . . . id='profile_ID' . . . .>` and `<offering . . . profile='profile_ID' . . . .>`) can be found when you run the `imcl listInstallationDirectories -verbose` command from the `eclipse/tools` subdirectory in the directory where you installed Installation Manager. It is the same as the package group's name.
- The offering ID (`<offering . . . id='offering_ID' . . . .>`) can be found in WebSphere Application Server product offerings for supported operating systems.
- The *version* is a specific version of the offering to install (8.5.5.20101025\_2108 for example). This specification is optional.

- If *version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
- If *version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
./imcl listAvailablePackages -repositories source_repository
```

- You can also specify *none*, *recommended* or *all* with the `installFixes` argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the `installFixes` option defaults to *all*.
  - If the offering version is specified, the `installFixes` option defaults to *none*.
-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install Liberty Repository assets, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later. Previous versions of Installation Manager do not have the option to install Liberty Repository assets. If you use a response file and did not update Installation Manager to Version 1.6.2 or later, the assets that you specify in the response file are ignored during installation.

If you want to install additional features, specify two extra data key elements in your response file. You can use either the symbolic name or the short name.

The following example installs the Portlet Container and Portlet Serving features using the symbolic name.


```
<data key='user.feature' value='com.ibm.websphere.appserver.portlet-2.0,,com.ibm.websphere.appserver.portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

The following example installs the Portlet Container and Portlet Serving features using the short name:

```
<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

**Note:** **8.5.5.3** If you are updating to Version 8.5.5.3 and previously installed Liberty Repository features but do not currently have a connection to the IBM WebSphere Liberty Repository, you cannot update using a response file. Instead, update the product by running the `imcl` command and specifying the `user.feature=""` parameter.


 **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features

The following example installs the Extended Programming Models using the `user.addon` parameter and the Portlet Container and Portlet Serving features using the `user.feature` parameter with short names:

```
<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.accept.license' value='true' />
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

 **8.5.5.6** You can also install assets from an instance of the Liberty Asset Repository Service or local directory-based repositories. For more information, see “Installing assets using Installation Manager” on page 872. Add the repository on `repository` elements. If Installation Manager does not recognize the repository, point directly to the `repository.config` file. When you install assets, the repositories are accessed in the order that you specify them.



```

<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85" />
<repository location="https://your_onprem_asset_repo_url" />
<repository location="D:\IBM\LocalAssetRepo" />

```

**8.5.5.8** `<repository location="D:\IBM\LocalAssetRepo2.zip" />` `</server>` By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, set the `user.useLibertyRepository` parameter to false:

```

<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portlet-serving-2.0' />
<data key='user.useLibertyRepository' value='false' />

```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- When updating a product, your response file must contain the features that were used in the initial install of the product; otherwise, your update will not occur as intended. If you do not add these features to your response file, Installation Manager assumes you are removing them.

### Uninstalling Liberty fix packs from distributed operating systems using the GUI:

You can roll back WebSphere Application Server Liberty Core to an earlier version using the Installation Manager GUI.

#### Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

**Important:** Installation Manager can install any fix-pack level of the product directly without installing the intermediate fix packs; in fact, Installation Manager installs the latest level by default. For example, you can skip fix-pack levels and go from Version 8.5.5.1 directly to Version 8.5.5.5. Keep in mind, however, that later you can not roll back to any level that was skipped. If you directly install to Version 8.5.5.5, for example, you cannot roll back to Version 8.5.5.4. If you skip from Version 8.5.5.1 to Version 8.5.5.5, you can only roll back to Version 8.5.5.1.

#### Procedure

1. Stop all servers on the WebSphere Application Server Liberty Core installation that is being modified.
2. Start Installation Manager.
3. Click **Roll Back**.
4. Select the package group to roll back.
5. Click **Next**.
6. Select the version to which you want to roll back under **IBM WebSphere Application Server Liberty Core**.
7. Click **Next**.
8. Review the summary information, and click **Roll Back**.
  - If the rollback is successful, the program displays a message indicating that the rollback is successful.
  - If the rollback is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

## Uninstalling Liberty fix packs from distributed operating systems using the command line:

You can roll back WebSphere Application Server Liberty Core to an earlier version using the Installation Manager command line.

### Before you begin

**Restriction:** In order to use this procedure, you must have Installation Manager Version 1.6 or later installed on your system.

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

### Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Log on to your system.
3. Stop all servers and applications on the installation that is being rolled back.
4. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
5. Use the `imcl` command to roll back the product.

AIX

HP-UX

Linux

Solaris

```
./imcl rollback offering_ID offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties property_key=value
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

Windows

```
imcl.exe rollback offering_ID offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties property_key=value
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

### Tips:

- The `offering_ID` is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.

- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to which to roll back (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the installation rolls back to the previously installed version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the installation rolls back to the specified earlier version of the offering and **no** interim fixes for that version are installed.
- If you previously installed Liberty Repository features and addons but do not have access to a Liberty repository when you rollback your installation, specify the following properties in the response file:
 

```
-properties user.addon="",user.feature=""
```

Specifying these properties enables the product to rollback and uninstalls all features and addons.

- **8.5.5.4** If you are rolling back from Version 8.5.5.4 to Version 8.5.5.3 and you have the Extended Programming Models installed, you might receive errors. To prevent errors, specify a properties parameter, `addon=""`; for example:

AIX

HP-UX

Linux

Solaris

```
./imcl rollback offering_ID offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties user.accept.license=true,user.addon=""
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

Windows

```
imcl.exe rollback offering_ID offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties user.accept.license=true,user.addon=""
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

For more information on using Installation Manager, read the IBM Installation Manager documentation.

6. Optional: List all installed packages to verify the roll back.

AIX

HP-UX

Linux

Solaris

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

## Uninstalling Liberty fix packs from distributed operating systems by using response files:

You can rollback WebSphere Application Server Liberty Core to an earlier version by using Installation Manager response files.

### Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

## Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When you create a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see Installation Manager Version 1.6 documentation. For more information on `-keyring` and `-password` options to store credentials in a keyring file, see Installation Manager Version 1.5 documentation.

2. Log on to your system.
3. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being rolled back.
4. Use a response file to roll back the product.

Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and roll back the product.

For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\rollback_response_file.xml
-log C:\temp\rollback_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/rollback_response_file.xml
-log /var/temp/rollback_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/rollback_response_file.xml
-log user_home/var/temp/rollback_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

**Note:** The program might write important post-installation instructions to standard output.

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

5. Optional: List all installed packages to verify the rollback.

**AIX HP-UX Linux Solaris**

```
./imcl listInstalledPackages -long
```

**Windows**

```
imcl.exe listInstalledPackages -long
```

**Windows**

## Example

The following is an example of a response file for rolling back the product to an earlier version.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
```

```

<repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85'/>
</server>
<profile id='WebSphere Liberty V8.5.5' installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
<rollback>
 <offering profile='WebSphere Liberty V8.5.5' id='com.ibm.websphere.liberty.v85' version='8.5.5.20101025_2108' />
</rollback>
</agent-input>

```

### Tips:

- The profile ID (<profile . . . id='profile\_ID' . . . .> and <offering . . . profile='profile\_ID' . . . .>) can be found when you run the `imcl listInstallationDirectories -verbose` command from the `eclipse/tools` subdirectory in the directory where you installed Installation Manager. It is the same as the package group's name.
- The offering ID (<offering . . . id='offering\_ID' . . . .>) can be found in WebSphere Application Server product offerings for supported operating systems.
- The *version* is a specific version of the offering to which to roll back (8.5.5.20101025\_2108 for example).

This specification is optional.

- If *version* is **not** specified, the installation rolls back to the previously installed version of the offering and **all** interim fixes for that version are installed.
- If *version* is specified, the installation rolls back to the specified earlier version of the offering and **no** interim fixes for that version are installed.
- If you previously installed Liberty Repository features and add-ons but do not have access to a Liberty repository when you rollback your installation, specify the following properties in the response file:

```

<data key='user.feature' value='' />
<data key='user.addon' value='' />

```

Specifying these properties enables the product to rollback and uninstalls all features and add-ons.

## Upgrading Liberty on distributed operating systems using the GUI

You can use Installation Manager GUI to upgrade WebSphere Application Server Liberty offerings on distributed operating systems.

### Before you begin

- Make sure that your Installation Manager preferences are pointing to web-based or local repositories that contain the appropriate upgrades for the offering.
- If an interim fix is already installed on the offering that you are trying to upgrade but that specific interim fix is not applicable to the offering to which you want to upgrade, the upgrade will be blocked by Installation Manager. An error message will be generated that indicates that the interim fix is not applicable to the offering. Remove the interim fix before upgrading.

### About this task

You can use Installation Manager to upgrade WebSphere Application Server Liberty offerings in the following paths:

- WebSphere Application Server Liberty Core to WebSphere Application Server Liberty
- WebSphere Application Server Liberty Core to WebSphere Application Server Liberty Network Deployment
- WebSphere Application Server Liberty - Express to WebSphere Application Server Liberty
- WebSphere Application Server Liberty to WebSphere Application Server Liberty Network Deployment
- WebSphere Application Server Liberty Trial to WebSphere Application Server Liberty
- WebSphere Application Server Liberty Core Trial to WebSphere Application Server Liberty Core
- WebSphere Application Server Liberty Network Deployment Trial to WebSphere Application Server Liberty Network Deployment

- WebSphere Application Server Liberty for Developers to WebSphere Application Server Liberty
- WebSphere Application Server Liberty for Developers to WebSphere Application Server Liberty Network Deployment

## Procedure

1. Stop all servers and applications on the WebSphere Application Server Liberty installation that is being upgraded.
2. Start Installation Manager.
3. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

4. Select the name of the Liberty offering to which you want to upgrade and the appropriate version, and click **Next**.
5. Accept the terms in the license agreements, and click **Next**.
6. Complete the following actions.
  - a. Select **Use the existing package group**.
  - b. Select the package group of the product that you want to upgrade.

**Important:** If you select an existing group that cannot be upgraded to the product that you are currently installing, an error will occur.

- c. Click **Next**.
7. Select any features that you want to include in the upgraded installation and click **Next**.
  8. Review the summary information, and click **Install**.
    - If the upgrade is successful, the program displays a message indicating that installation is successful.
- Note:** The program might also display important post-installation instructions as well.
- If the installation is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
  10. Click **File > Exit** to close Installation Manager.

## Example

Here is an example of how to upgrade a WebSphere Application Server Liberty offering on distributed operating systems using the command line.

- Log on to your system.
- Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
- List the installed packages.

AIX HP-UX Linux Solaris

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

- Use the `imcl install` command to upgrade the installation to a different edition in the same installation location:

AIX HP-UX Linux Solaris

```
./imcl install com.ibm.websphere.liberty.v85
-repositories source_repository -installationDirectory /opt/IBM/WebSphere/Liberty -acceptLicense -showProgress
```

## Windows

```
imcl install com.ibm.websphere.liberty.v85
-repositories source_repository -installationDirectory C:\Program Files\IBM\WebSphere\Liberty -acceptLicense -showProg
```

## Installing and uninstalling SDK Java Technology Edition Version 7.0 or 7.1 for Liberty on distributed operating systems

You can install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using Installation Manager.

### About this task

Perform one of these procedures to install, update, roll back, or uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using Installation Manager.

#### 8.5.5.2

You can download and install IBM WebSphere SDK Java Technology Edition Version 7.1. This version is only available from the web. This version is not available on the physical media. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

### Procedure

- “Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI” on page 736
- “Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using response files” on page 740
- “Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the command line” on page 738
- “Installing fix packs on IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI” on page 744
- “Uninstalling fix packs from IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI” on page 745
- “Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI” on page 746
- “Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using response files” on page 747
- “Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the command line” on page 746

### Results

#### Notes on logging and tracing:

- An easy way to view the logs is to open Installation Manager and go to **File > View Log**. An individual log file can be opened by selecting it in the table and then clicking the **Open log file** icon.
- Logs are located in the logs directory of Installation Manager's application data location. For example:

– **Windows** **Administrative installation:**

```
C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager\logs
```

– **Windows** **Non-administrative installation:**

```
C:\Documents and Settings\user_name\Application Data\IBM\Installation Manager\logs
```

– **AIX** **HP-UX** **Linux** **Solaris** **Administrative installation:**

```
/var/ibm/InstallationManager/logs
```

– **AIX** **HP-UX** **Linux** **Solaris** **Non-administrative installation:**

`user_home/var/ibm/InstallationManager/logs`

- The main log files are time-stamped XML files in the logs directory, and they can be viewed using any standard web browser.
- The `log.properties` file in the logs directory specifies the level of logging or tracing that Installation Manager uses.

### Notes on troubleshooting:

- **HP-UX** By default, some HP-UX systems are configured to not use DNS to resolve host names. This could result in Installation Manager not being able to connect to an external repository.

You can ping the repository, but `nslookup` does not return anything.

Work with your system administrator to configure your machine to use DNS, or use the IP address of the repository.

- For more information on using Installation Manager, read the IBM Installation Manager Information Center.

Read the release notes to learn more about the latest version of Installation Manager. To access the release notes, complete the following task:

– **Windows** Click **Start > Programs > IBM Installation Manager > Release Notes**.

– **AIX** **HP-UX** **Linux** **Solaris** Go to the documentation subdirectory in the directory where Installation Manager is installed, and open the `readme.html` file.

### Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI:

You can install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the Installation Manager GUI.

#### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

#### 8.5.5.2

You can download and install IBM WebSphere SDK Java Technology Edition Version 7.1. This version is only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

#### Procedure

1. Start Installation Manager.



**Tip:** AIX HP-UX Linux Solaris You can start Installation Manager in group mode with the `./IBMIM` command.

- Group mode allows users to share packages in a common location and manage them with the same instance of Installation Manager.
- For more information on using group mode, read the Group mode road maps in the IBM Installation Manager Information Center.

2. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

3. Perform the appropriate actions.

- a. Select **IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty** and the appropriate version.

If you already have IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty installed on a WebSphere Application Server Liberty Core installation on your system, a message displays indicating that IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty is already installed. To install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty on an installation where it is not yet installed, click **Continue**.

- b. Select the fixes to install.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

- c. Click **Next**.

**Note:** If you do not have and have not selected a Version 8.5 installation of one the following offerings on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, an error dialog displays.

- IBM WebSphere Application Server Liberty
- IBM WebSphere Application Server Liberty Trial
- IBM WebSphere Application Server Liberty Core
- IBM WebSphere Application Server Liberty Core Trial
- IBM WebSphere Application Server Liberty - Express
- IBM WebSphere Application Server Liberty Network Deployment
- IBM WebSphere Application Server Liberty Network Deployment Trial
- IBM WebSphere Application Server Liberty for Developers
- IBM WebSphere Application Server Liberty for Developers (ILAN)

Click **OK**, select an appropriate offering on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, and click **Next**.

4. Select the appropriate installation on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, and click **Next**.

**Note:** The bit architecture that will be installed, 32 bit or 64 bit, will match the bit selection made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden using the `cic.selector.arch` property.

5. Click **Next**.

6. Review the summary information, and click **Install**.

- If the installation is successful, the program displays a message indicating that installation is successful.

**Note:** The program might also display important post-installation instructions as well.

- If the installation is not successful, click **View Log File** to troubleshoot the problem.

7. Click **Finish**.

8. Click **File > Exit** to close Installation Manager.

### Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the command line:

You can install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the Installation Manager command line.

#### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

#### 8.5.5.2

You can download and install IBM WebSphere SDK Java Technology Edition Version 7.1. This version is only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

#### Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Log on to your system.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.

#### 4. Verify that the repository is available.

Windows

```
imcl.exe listAvailablePackages -repositories path_to_repository
```

AIX

HP-UX

Linux

Solaris

```
./imcl listAvailablePackages -repositories path_to_repository
```

You should see one or more levels of the offering.

#### 5. Use the **imcl** command to install the offering.

Windows

```
imcl.exe install com.ibm.websphere.liberty.IBMJAVA.v70_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

8.5.5.2

```
imcl.exe install com.ibm.websphere.liberty.IBMJAVA.v71_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

AIX

HP-UX

Linux

Solaris

```
./imcl install com.ibm.websphere.liberty.IBMJAVA.v70_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

8.5.5.2

```
./imcl install com.ibm.websphere.liberty.IBMJAVA.v71_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

The value of the **-installationDirectory** parameter should be the location of an existing Liberty installation. The SDK offering will be installed into the `<liberty_home>\java\java_1.7_32` or `<liberty_home>\java\java_1.7_64` folder, depending on whether the installation location was configured to be 32-bit or 64-bit when the Liberty offering was initially installed.

**Note:** The bit architecture that will be installed, 32 bit or 64 bit, will match the bit selection made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden using the `cic.selector.arch` property.

#### Tips:

- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.0.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the `-installFixes` argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the `-installFixes` option defaults to all.
  - If the offering version is specified, the `-installFixes` option defaults to none.

- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `product_name/lafiles` subdirectory of the installation image or repository for this product.
- The program might write important post-installation instructions to standard output.

For more information on using the `imcl` command, see the IBM Installation Manager Information Center.

### Installing IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using response files:

You can install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using Installation Manager response files.

#### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

#### About this task

Using Installation Manager, you can work with response files to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

#### 8.5.5.2

You can download and install IBM WebSphere SDK Java Technology Edition Version 7.1. This version and later versions are only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

#### Procedure

1. Optional: **Record a response file to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:** On one of your systems, perform the following actions to record a response file that will install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty.
  - a. From a command line, change to the `eclipse` subdirectory in the directory where you installed Installation Manager.
  - b. Start Installation Manager from the command line using the `-record` option.

For example:

- **Windows Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\install_response_file.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/install_response_file.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/install_response_file.xml
```

**Tip:** When you record a new response file, you can specify the `-skipInstall` parameter. Using this parameter has the following benefits:

- No files are actually installed, and this speeds up the recording.
- If you use a temporary data location with the `-skipInstall` parameter, Installation Manager writes the installation registry to the specified data location while recording. When you start Installation Manager again without the `-skipInstall` parameter, you then can use your response file to install against the real installation registry.

The `-skipInstall` operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

For more information, read the IBM Installation Manager Information Center.

- c. Add the appropriate repositories to your Installation Manager preferences.

- 1) Click **File > Preferences**.

- 2) Select **Repositories**.

- 3) Perform the following actions for each repository:

- a) Click **Add Repository**.

- b) Enter the path to the `repository.config` file in the remote web-based repository or the local directory into which you unpacked the repository files.

For example:

- Remote repositories:

[https://downloads.mycorp.com:8080/WAS\\_85\\_repository](https://downloads.mycorp.com:8080/WAS_85_repository)

or **8.5.5.1**

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v70>

**8.5.5.2**

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v71>

- Local repositories:

– **Windows** `C:\repositories\jdk7\local-repositories`

– **AIX** **HP-UX** **Linux** **Solaris** `/var/repositories/jdk7/local-repositories`

- c) Click **OK**.

- 4) Click **Apply**.

- 5) Click **OK**.

- d. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

- e. Perform the appropriate actions.

- 1) Select **IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty**.

If you already have IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty installed on a WebSphere Application Server Liberty Core installation on your system

in the targeted location, a message displays indicating that IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty is already installed. To install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 on a WebSphere Application Server Liberty Core installation where it is not yet installed, click **Continue**.

- 2) Select the fixes to install.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

- 3) Click **Next**.

**Note:** If you do not have and have not selected an installation of one the following offerings on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, an error dialog displays.

- IBM WebSphere Application Server Liberty
- IBM WebSphere Application Server Liberty Trial
- IBM WebSphere Application Server Liberty Core
- IBM WebSphere Application Server Liberty Core Trial
- IBM WebSphere Application Server Liberty - Express
- IBM WebSphere Application Server Liberty Network Deployment
- IBM WebSphere Application Server Liberty Network Deployment Trial
- IBM WebSphere Application Server Liberty for Developers
- IBM WebSphere Application Server Liberty for Developers (ILAN)

Click **OK**, select an appropriate offering on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, and click **Next**.

- f. Select the appropriate WebSphere Application Server Liberty Core installation on which to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, and click **Next**.

**Note:** The bit architecture that will be installed, 32 bit or 64 bit, will match the bit selection made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden using the `cic.selector.arch` property.

- g. Click **Next**.

- h. Review the summary information, and click **Install**.

- If the installation is successful, the program displays a message indicating that installation is successful.

**Note:** The program might also display important post-installation instructions as well.

- If the installation is not successful, click **View Log File** to troubleshoot the problem.

- i. Click **Finish**.

- j. Click **File > Exit** to close Installation Manager.

- k. Optional: If you are using an authenticated remote repository, create a credential-storage file for silent installation.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6

documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

## 2. Use the response files to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:

- a. **Optional: Use the response file to install the credential-storage file:** Go to a command line on each of the systems on which you want to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the credential-storage file.

For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\credentialstorage_response_file.xml
-log C:\temp\credentialstorage_log.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/credentialstorage_response_file.xml
-log /var/temp/credentialstorage_log.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/credentialstorage_response_file.xml
-log user_home/var/temp/credentialstorage_log.xml
```

- b. **Use the response file to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:** Go to a command line on each of the systems on which you want to install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty.

For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\install_response_file.xml
-log C:\temp\install_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/install_response_file.xml
-log /var/temp/install_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/install_response_file.xml
-log user_home/var/temp/install_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

### Notes:

- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `product_name/lafiles` subdirectory of the installation image or repository for this product.
- The program might write important post-installation instructions to standard output.

Read the IBM Installation Manager Information Center.

### Example

**Windows 8.5.5.1** The following is an example of a response file for installing IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean="true" temporary="true">
<server>
```

```

<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v70" />
</server>
<install modify='false'>
<offering id='com.ibm.websphere.liberty.IBMJAVA.v70'
 profile='WebSphere Liberty V8.5' installFixes='none' />
</install>
<profile id='WebSphere Liberty V8.5' installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>

```

Windows

8.5.5.2

The following is an example of a response file for installing IBM WebSphere SDK Java Technology Edition Version 7.1 for Liberty.

```

<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean="true" temporary="true">
<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v71" />
</server>
<install modify='false'>
<offering id='com.ibm.websphere.liberty.IBMJAVA.v71'
 profile='WebSphere Liberty V8.5' installFixes='none' />
</install>
<profile id='WebSphere Liberty V8.5' installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>

```

## Installing fix packs on IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI:

You can update IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty to a later version using the Installation Manager GUI.

### Before you begin

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the IBM Installation Manager Information Center.

### About this task

**Note:** For information on installing and removing fix packs for WebSphere Application Server Liberty Core on distributed operating systems using the Installation Manager command line, read the following articles in this information center:

- “Installing Liberty fix packs on distributed operating systems using the command line” on page 720
- “Uninstalling Liberty fix packs from distributed operating systems using the command line” on page 730

### Procedure

1. Start Installation Manager.
2. Click **Update**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.

3. Select the package group to update.

**Tip:** If you select **Update all**, Installation Manager will search all of the added and predefined repositories for recommended updates to all of the package groups that it has installed.

- Use this feature only if you have full control over which fixes are contained in the targeted repositories.
  - If you create and point to a set of custom repositories that include only the specific recommended fixes that you want to install, you should be able to use this feature confidently.



- If you enable searching service repositories or install fixes directly from other live web-based repositories, you might not want to select this option so that you can select only the fixes that you want to install for each offering on subsequent panels.
  - If you select **Update all**, Installation Manager will install only the recommended updates to all of the package groups; it will not allow you to select non-recommended fixes for installation. If you want to install non-recommended fixes, perform the following actions:
    - a. On this panel, clear the **Update all** check box and select an offering to update.
    - b. On the next panel, clear the option to show only recommended fixes and then select the fixes that you want to install.
4. Click **Next**.
  5. Select the version to which you want to update under **IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty**.
  6. Click **Next**.
  7. Accept the terms in the license agreements, and click **Next**.
  8. Review the summary information, and click **Update**.
    - If the installation is successful, the program displays a message indicating that installation is successful.
    - If the installation is not successful, click **View Log File** to troubleshoot the problem.
  9. Click **Finish**.
  10. Click **File > Exit** to close Installation Manager.

### Uninstalling fix packs from IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI:

You can roll back IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty to an earlier version using the Installation Manager GUI.

#### Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

#### About this task

**Note:** For information on installing and removing fix packs for WebSphere Application Server Liberty Core on distributed operating systems using the Installation Manager command line, read the following articles in this information center:

- “Installing Liberty fix packs on distributed operating systems using the command line” on page 720
- “Uninstalling Liberty fix packs from distributed operating systems using the command line” on page 730

#### Procedure

1. Stop all servers on the installation that is being modified.
2. Start Installation Manager.
3. Click **Roll Back**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.

4. Select the package group to roll back.

5. Click **Next**.
6. Select the version to which you want to roll back under **IBM WebSphere SDK Java Technology Edition Version 6.0 for Liberty**.
7. Click **Next**.
8. Review the summary information, and click **Roll Back**.
  - If the rollback is successful, the program displays a message indicating that the rollback is successful.
  - If the rollback is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

### **Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the GUI:**

Use the Installation Manager GUI to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty.

#### **Before you begin**

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty will not function if it is uninstalled.

#### **Procedure**

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. Start Installation Manager.
4. Click **Uninstall**.
5. In the **Uninstall Packages** window, perform the following actions.
  - a. Select **IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty** and the appropriate version.
  - b. Click **Next**.
6. Review the summary information.
7. Click **Uninstall**.
  - If the uninstallation is successful, the program displays a message that indicates success.
  - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
8. Click **Finish**.
9. Click **File > Exit** to close Installation Manager.

### **Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the command line:**

You can uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using the Installation Manager command line.

#### **Before you begin**

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty will not function if it is uninstalled.

## Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
4. Use the `imcl` command to uninstall the offering.

Windows

```
imcl.exe uninstall com.ibm.websphere.liberty.IBMJAVA.v70
-installationDirectory installation_directory
```

AIX

HP-UX

Linux

Solaris

```
./imcl uninstall com.ibm.websphere.liberty.IBMJAVA.v70
-installationDirectory installation_directory
```

Go to the IBM Installation Manager Information Center.

## Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using response files:

You can uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty using Installation Manager response files.

### Before you begin

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty will not function if it is uninstalled.

**Optional:** Perform or record the installation of Installation Manager and installation of IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty to a temporary installation registry on one of your systems so that you can use this temporary registry to record the uninstallation without using the standard registry where Installation Manager is installed.

### About this task

Using Installation Manager, you can work with response files to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

## Procedure

1. **Optional: Record a response file to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:** On one of your systems, perform the following actions to record a response file that will uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:
  - a. From a command line, change to the `eclipse` subdirectory in the directory where you installed Installation Manager.
  - b. Start Installation Manager from the command line using the `-record` option.

For example:

- **Windows Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\uninstall_response_file.xml
```

AIX

HP-UX

Linux

Solaris

**Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/uninstall_response_file.xml
```

AIX

HP-UX

Linux

Solaris

**Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/uninstall_response_file.xml
```

**Tip:** If you choose to use the `-skipInstall` parameter with a temporary installation registry created as described in *Before you begin*, Installation Manager uses the temporary installation registry while recording the response file. It is important to note that when the `-skipInstall` parameter is specified, no packages are installed or uninstalled. All of the actions that you perform in Installation Manager simply update the installation data that is stored in the specified temporary registry. After the response file is generated, it can be used to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, removing the files and updating the standard installation registry.

The `-skipInstall` operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

For more information, read the IBM Installation Manager Information Center.

- c. Click **Uninstall**.
  - d. In the **Uninstall Packages** window, perform the following actions.
    - 1) Select **IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty** and the appropriate version.
    - 2) Click **Next**.
  - e. Review the summary information.
  - f. Click **Uninstall**.
    - If the uninstallation is successful, the program displays a message that indicates success.
    - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
  - g. Click **Finish**.
  - h. Click **File > Exit** to close Installation Manager.
2. **Use the response file to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty:** From a command line on each of the systems from which you want to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager and use the response file that you created to uninstall IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty. For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\uninstall_response_file.xml
-log C:\temp\uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/uninstall_response_file.xml
-log /var/temp/uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/uninstall_response_file.xml
-log user_home/var/temp/uninstall_log.xml
```

Go to the IBM Installation Manager Information Center.

3. **Optional: List all installed packages to verify the uninstallation.**

**AIX HP-UX Linux Solaris**

```
./imcl listInstalledPackages
```

**Windows**

```
imcl listInstalledPackages
```

## Example

The following is an example of a response file for uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 or 7.1 for Liberty.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>
<uninstall modify='false'>
<offering id='com.ibm.websphere.liberty.IBMJAVA.v70'
 profile='WebSphere Liberty V8.5' />
</uninstall>
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

## Installing and uninstalling SDK Java Technology Edition Version 8.0 for Liberty on distributed operating systems

8.5.5.5

You can install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using Installation Manager.

### About this task

Perform one of these procedures to install, update, roll back, or uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using Installation Manager.

You can download and install IBM WebSphere SDK Java Technology Edition Version 8.0. This version is only available from the web. This version is not available on the physical media. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

### Procedure

- Install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI.
- Install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using response files.
- Install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the command line.
- Install fix packs on IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI.
- Uninstall fix packs on IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI.
- Uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI.
- Uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using response files.
- Uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the command line.

### Results

#### Notes on logging and tracing:

- An easy way to view the logs is to open Installation Manager and go to **File > View Log**. An individual log file can be opened by selecting it in the table and then clicking the **Open log file** icon.
- Logs are located in the logs directory of Installation Manager's application data location. For example:

— **Windows Administrative installation:**

C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager\logs

– **Windows** **Non-administrative installation:**

C:\Documents and Settings\*user\_name*\Application Data\IBM\Installation Manager\logs

– **AIX** **HP-UX** **Linux** **Solaris** **Administrative installation:**

/var/ibm/InstallationManager/logs

– **AIX** **HP-UX** **Linux** **Solaris** **Non-administrative installation:**

*user\_home*/var/ibm/InstallationManager/logs

- The main log files are time-stamped XML files in the logs directory, which you can view using any standard web browser.
- The log.properties file in the logs directory specifies the level of logging or tracing that Installation Manager uses.

### Notes on troubleshooting:

- **HP-UX** By default, some HP-UX systems are configured to not use DNS to resolve host names. This can result in Installation Manager not being able to connect to an external repository.

You can ping the repository, but nslookup does not return anything.

Work with your system administrator to configure your machine to use DNS, or use the IP address of the repository.

- For more information on using Installation Manager, read the IBM Installation Manager Information Center.

Read the release notes to learn more about the latest version of Installation Manager. To access the release notes, complete the following task:

– **Windows** Click **Start > Programs > IBM Installation Manager > Release Notes**.

– **AIX** **HP-UX** **Linux** **Solaris** Go to the documentation subdirectory in the directory where Installation Manager is installed, and open the readme.html file.

## Installing IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI:

8.5.5.5

You can install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty by using the Installation Manager GUI.

### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

You can download and install IBM WebSphere SDK Java Technology Edition Version 8.0. This version is only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

**Note:** You can locate the installation files here IBM Support: Fix central .

## Procedure

### 1. Start Installation Manager.

**Tip:** AIX HP-UX Linux Solaris You can start Installation Manager in group mode with the `./IBMIM` command.

- Group mode allows users to share packages in a common location and manage them with the same instance of Installation Manager.
- For more information on using group mode, read the Group mode road maps in the IBM Installation Manager Information Center.

### 2. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

### 3. Perform the appropriate actions.

- a. Select **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty** and the appropriate version.

If you already have IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty installed on a WebSphere Application Server Liberty Core installation on your system, a message displays indicating that IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty is already installed. To install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty on an installation where it is not yet installed, click **Continue**.

- b. Select the fixes to install.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

- c. Click **Next**.

**Note:** If you do not have and have not selected a Version 8.5 installation of one the following offerings on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, an error dialog displays.

- IBM WebSphere Application Server Liberty
- IBM WebSphere Application Server Liberty Trial
- IBM WebSphere Application Server Liberty Core
- IBM WebSphere Application Server Liberty Core Trial
- IBM WebSphere Application Server Liberty - Express
- IBM WebSphere Application Server Liberty Network Deployment
- IBM WebSphere Application Server Liberty Network Deployment Trial
- IBM WebSphere Application Server Liberty for Developers
- IBM WebSphere Application Server Liberty for Developers (ILAN)

Click **OK**, select an appropriate offering on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, and click **Next**.

### 4. Select the appropriate installation on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, and click **Next**.

**Note:** The bit architecture that is installed, 32 bit or 64 bit, matches the bit selection that is made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection

is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden by using the `cic.selector.arch` property.

5. Click **Next**.
6. Review the summary information, and click **Install**.
  - If the installation is successful, the program displays a message that says that installation is successful.

**Note:** The program might also display important post-installation instructions.

- If the installation is not successful, click **View Log File** to troubleshoot the problem.
7. Click **Finish**.
  8. Click **File > Exit** to close Installation Manager.

### Installing IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the command line: 8.5.5.5

You can install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the Installation Manager command line.

#### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

You can download and install IBM WebSphere SDK Java Technology Edition Version 8.0. This version is only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

#### Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For



more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Log on to your system.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
4. Verify that the repository is available.

Windows

```
imcl.exe listAvailablePackages -repositories path_to_repository
```

AIX

HP-UX

Linux

Solaris

```
./imcl listAvailablePackages -repositories path_to_repository
```

You should see one or more levels of the offering.

5. Use the **imcl** command to install the offering.

Windows

```
imcl.exe install com.ibm.websphere.liberty.IBMJAVA.v80_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

AIX

HP-UX

Linux

Solaris

```
./imcl install com.ibm.websphere.liberty.IBMJAVA.v80_offering_version
-repositories source_repository
-installationDirectory installation_directory
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

The value of the **-installationDirectory** parameter should be the location of an existing Liberty installation. The SDK offering will be installed into the `<liberty_home>\java\java_1.8_32` or `<liberty_home>\java\java_1.8_64` folder, depending on whether the installation location was configured to be 32-bit or 64-bit when the Liberty offering was initially installed.

**Note:** The bit architecture that will be installed, 32 bit or 64 bit, will match the bit selection made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden using the `cic.selector.arch` property.

#### Tips:

- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.0.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the **-installFixes** argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the **-installFixes** option defaults to all.
  - If the offering version is specified, the **-installFixes** option defaults to none.
- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `product_name/lafiles` subdirectory of the installation image or repository for this product.
- The program might write important post-installation instructions to standard output.

For more information on using the **imcl** command, see the IBM Installation Manager Information Center.

## Installing IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using response files:

8.5.5.5

You can install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using Installation Manager response files.

### Before you begin

1. Prepare your system as described in “Installing Installation Manager and preparing to install Liberty” on page 694.
2. Install one the following offerings:
  - IBM WebSphere Application Server Liberty
  - IBM WebSphere Application Server Liberty Trial
  - IBM WebSphere Application Server Liberty Core
  - IBM WebSphere Application Server Liberty Core Trial
  - IBM WebSphere Application Server Liberty - Express
  - IBM WebSphere Application Server Liberty Network Deployment
  - IBM WebSphere Application Server Liberty Network Deployment Trial
  - IBM WebSphere Application Server Liberty for Developers
  - IBM WebSphere Application Server Liberty for Developers (ILAN)

### About this task

Using Installation Manager, you can work with response files to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

You can download and install IBM WebSphere SDK Java Technology Edition Version 8.0. This version and later versions are only available from the web. You have the following installation options:

- Download the installation files from the IBM Fix Central website and use a local installation.
- Access the live repositories and use your IBM Software ID for a web-based installation.

### Procedure

1. Optional: **Record a response file to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:** On one of your systems, perform the following actions to record a response file that will install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.
  - a. From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.
  - b. Start Installation Manager from the command line using the -record option.

For example:

- **Windows Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\install_response_file.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/install_response_file.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/install_response_file.xml
```

**Tip:** When you record a new response file, you can specify the `-skipInstall` parameter. Using this parameter has the following benefits:

- No files are actually installed, and this speeds up the recording.
- If you use a temporary data location with the `-skipInstall` parameter, Installation Manager writes the installation registry to the specified data location while recording. When you start Installation Manager again without the `-skipInstall` parameter, you then can use your response file to install against the real installation registry.

The `-skipInstall` operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

For more information, read the IBM Installation Manager Information Center.

c. Add the appropriate repositories to your Installation Manager preferences.

1) Click **File > Preferences**.

2) Select **Repositories**.

3) Perform the following actions for each repository:

a) Click **Add Repository**.

b) Enter the path to the `repository.config` file in the remote web-based repository or the local directory into which you unpacked the repository files.

For example:

- Remote repositories:

`https://downloads.mycorp.com:8080/WAS_85_repository`

or

`http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v80`

- Local repositories:

– **Windows** `C:\repositories\jdk8\local-repositories`

– **AIX** **HP-UX** **Linux** **Solaris** `/var/repositories/jdk8/local-repositories`

c) Click **OK**.

4) Click **Apply**.

5) Click **OK**.

d. Click **Install**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

e. Perform the appropriate actions.

1) Select **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty**.

If you already have IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty installed on a WebSphere Application Server Liberty Core installation on your system in the targeted location, a message displays indicating that IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty is already installed. To install IBM WebSphere SDK Java Technology Edition Version 8.0 on a WebSphere Application Server Liberty Core installation where it is not yet installed, click **Continue**.

2) Select the fixes to install.

Any recommended fixes are selected by default.

If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

3) Click **Next**.

**Note:** If you do not have and have not selected an installation of one of the following offerings on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, an error dialog displays.

- IBM WebSphere Application Server Liberty
- IBM WebSphere Application Server Liberty Trial
- IBM WebSphere Application Server Liberty Core
- IBM WebSphere Application Server Liberty Core Trial
- IBM WebSphere Application Server Liberty - Express
- IBM WebSphere Application Server Liberty Network Deployment
- IBM WebSphere Application Server Liberty Network Deployment Trial
- IBM WebSphere Application Server Liberty for Developers
- IBM WebSphere Application Server Liberty for Developers (ILAN)

Click **OK**, select an appropriate offering on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, and click **Next**.

- f. Select the appropriate WebSphere Application Server Liberty Core installation on which to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, and click **Next**.

**Note:** The bit architecture that will be installed, 32 bit or 64 bit, will match the bit selection made during the installation of the Liberty offering. For GUI installations of Liberty offerings, this selection is made on the Location panel. For command-line and response-file installations of Liberty offerings, the default can be overridden using the `cic.selector.arch` property.

- g. Click **Next**.
- h. Review the summary information, and click **Install**.
  - If the installation is successful, the program displays a message indicating that installation is successful.

**Note:** The program might also display important post-installation instructions as well.

- If the installation is not successful, click **View Log File** to troubleshoot the problem.
- i. Click **Finish**.
- j. Click **File > Exit** to close Installation Manager.
- k. Optional: If you are using an authenticated remote repository, create a credential-storage file for silent installation.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

## 2. Use the response files to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:

- a. Optional: **Use the response file to install the credential-storage file:** Go to a command line on each of the systems on which you want to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the credential-storage file.

For example:

-  **Administrator or non-administrator:**

```
imcl.exe
input C:\temp\credentialstorage_response_file.xml
-log C:\temp\credentialstorage_log.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./imcl
input /var/temp/credentialstorage_response_file.xml
-log /var/temp/credentialstorage_log.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./imcl
input user_home/var/temp/credentialstorage_response_file.xml
-log user_home/var/temp/credentialstorage_log.xml
```

- b. **Use the response file to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:** Go to a command line on each of the systems on which you want to install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, change to the eclipse/tools subdirectory in the directory where you installed Installation Manager, and install IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.

For example:

- **Windows** **Administrator or non-administrator:**

```
imcl.exe
input C:\temp\install_response_file.xml
-log C:\temp\install_log.xml
-secureStorageFile C:\IM\credential.store -masterPasswordFile C:\IM\master_password_file.txt
```

- **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./imcl
input /var/temp/install_response_file.xml
-log /var/temp/install_log.xml
-secureStorageFile /var/IM/credential.store -masterPasswordFile /var/IM/master_password_file.txt
```

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./imcl
input user_home/var/temp/install_response_file.xml
-log user_home/var/temp/install_log.xml
-secureStorageFile user_home/var/IM/credential.store -masterPasswordFile user_home/var/IM/master_password_file.txt
```

### Notes:

- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the lafiles or *product\_name/lafiles* subdirectory of the installation image or repository for this product.
- The program might write important post-installation instructions to standard output.

Read the IBM Installation Manager Information Center.

### Example

**Windows** The following is an example of a response file for installing IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean="true" temporary="true">
<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v80" />
</server>
<install modify='false'>
<offering id='com.ibm.websphere.liberty.IBMJAVA.v80'
profile='WebSphere Liberty V8.5' installFixes='none' />
</install>
<profile id='WebSphere Liberty V8.5' installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

### Installing fix packs on IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI: **8.5.5.5**

You can update IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty to a later version using the Installation Manager GUI.

## Before you begin

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the IBM Installation Manager Information Center.

## About this task

**Note:** For information on installing and removing fix packs for WebSphere Application Server Liberty Core on distributed operating systems using the Installation Manager command line, read the following articles in this information center:

- “Installing Liberty fix packs on distributed operating systems using the command line” on page 720
- “Uninstalling Liberty fix packs from distributed operating systems using the command line” on page 730

## Procedure

1. Start Installation Manager.
2. Click **Update**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.

3. Select the package group to update.

**Tip:** If you select **Update all**, Installation Manager will search all of the added and predefined repositories for recommended updates to all of the package groups that it has installed.

- Use this feature only if you have full control over which fixes are contained in the targeted repositories.
  - If you create and point to a set of custom repositories that include only the specific recommended fixes that you want to install, you should be able to use this feature confidently.
  - If you enable searching service repositories or install fixes directly from other live web-based repositories, you might not want to select this option so that you can select only the fixes that you want to install for each offering on subsequent panels.
- If you select **Update all**, Installation Manager will install only the recommended updates to all of the package groups; it will not allow you to select non-recommended fixes for installation. If you want to install non-recommended fixes, perform the following actions:
  - a. On this panel, clear the **Update all** check box and select an offering to update.
  - b. On the next panel, clear the option to show only recommended fixes and then select the fixes that you want to install.

4. Click **Next**.
5. Select the version to which you want to update under **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty**.
6. Click **Next**.
7. Accept the terms in the license agreements, and click **Next**.
8. Review the summary information, and click **Update**.
  - If the installation is successful, the program displays a message indicating that installation is successful.
  - If the installation is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

## Uninstalling fix packs from IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI: 8.5.5.5

You can roll back IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty to an earlier version using the Installation Manager GUI.

### Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

### About this task

**Note:** For information on installing and removing fix packs for WebSphere Application Server Liberty Core on distributed operating systems using the Installation Manager command line, read the following articles in this information center:

- “Installing Liberty fix packs on distributed operating systems using the command line” on page 720
- “Uninstalling Liberty fix packs from distributed operating systems using the command line” on page 730

### Procedure

1. Stop all servers on the installation that is being modified.
2. Start Installation Manager.
3. Click **Roll Back**.

**Note:** If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.

4. Select the package group to roll back.
5. Click **Next**.
6. Select the version to which you want to roll back under **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty**.
7. Click **Next**.
8. Review the summary information, and click **Roll Back**.
  - If the rollback is successful, the program displays a message indicating that the rollback is successful.
  - If the rollback is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

## Uninstalling IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the GUI:

Use the Installation Manager GUI to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.

### Before you begin

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty will not function if it is uninstalled.

## Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. Start Installation Manager.
4. Click **Uninstall**.
5. In the **Uninstall Packages** window, perform the following actions.
  - a. Select **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty** and the appropriate version.
  - b. Click **Next**.
6. Review the summary information.
7. Click **Uninstall**.
  - If the uninstallation is successful, the program displays a message that indicates success.
  - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
8. Click **Finish**.
9. Click **File > Exit** to close Installation Manager.

## Uninstalling IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the command line: 8.5.5.5

You can uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using the Installation Manager command line.

### Before you begin

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty will not function if it is uninstalled.

## Procedure

1. Log on to your system.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
4. Use the `imcl` command to uninstall the offering.

Windows

```
imcl.exe uninstall com.ibm.websphere.liberty.IBMJAVA.v80
-installationDirectory installation_directory
```

AIX

HP-UX

Linux

Solaris

```
./imcl uninstall com.ibm.websphere.liberty.IBMJAVA.v80
-installationDirectory installation_directory
```

Go to the IBM Installation Manager Information Center.

## Uninstalling IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using response files: 8.5.5.5

You can uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty using Installation Manager response files.



## Before you begin

Make sure that no commands or server profiles are using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty before uninstalling it. Server profiles using IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty will not function if it is uninstalled.

**Optional:** Perform or record the installation of Installation Manager and installation of IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty to a temporary installation registry on one of your systems so that you can use this temporary registry to record the uninstallation without using the standard registry where Installation Manager is installed.

## About this task

Using Installation Manager, you can work with response files to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

## Procedure

1. **Optional: Record a response file to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:** On one of your systems, perform the following actions to record a response file that will uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:
  - a. From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.
  - b. Start Installation Manager from the command line using the `-record` option.

For example:

-  **Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
-record C:\temp\uninstall_response_file.xml
```

-     **Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
-record /var/temp/uninstall_response_file.xml
```

-     **Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
-record user_home/var/temp/uninstall_response_file.xml
```

**Tip:** If you choose to use the `-skipInstall` parameter with a temporary installation registry created as described in *Before you begin*, Installation Manager uses the temporary installation registry while recording the response file. It is important to note that when the `-skipInstall` parameter is specified, no packages are installed or uninstalled. All of the actions that you perform in Installation Manager simply update the installation data that is stored in the specified temporary registry. After the response file is generated, it can be used to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, removing the files and updating the standard installation registry.

The `-skipInstall` operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

For more information, read the IBM Installation Manager Information Center.

- c. Click **Uninstall**.
- d. In the **Uninstall Packages** window, perform the following actions.
  - 1) Select **IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty** and the appropriate version.

2) Click **Next**.

e. Review the summary information.

f. Click **Uninstall**.

- If the uninstallation is successful, the program displays a message that indicates success.
- If the uninstallation is not successful, click **View log** to troubleshoot the problem.

g. Click **Finish**.

h. Click **File > Exit** to close Installation Manager.

2. **Use the response file to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty:** From a command line on each of the systems from which you want to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager and use the response file that you created to uninstall IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.

For example:

- **Windows Administrator or non-administrator:**

```
imcl.exe
input C:\temp\uninstall_response_file.xml
-log C:\temp\uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Administrator:**

```
./imcl
input /var/temp/uninstall_response_file.xml
-log /var/temp/uninstall_log.xml
```

- **AIX HP-UX Linux Solaris Non-administrator:**

```
./imcl
input user_home/var/temp/uninstall_response_file.xml
-log user_home/var/temp/uninstall_log.xml
```

Go to the IBM Installation Manager Information Center.

3. **Optional: List all installed packages to verify the uninstallation.**

**AIX HP-UX Linux Solaris**

```
./imcl listInstalledPackages
```

**Windows**

```
imcl listInstalledPackages
```

**Windows**

## Example

The following is an example of a response file for uninstalling IBM WebSphere SDK Java Technology Edition Version 8.0 for Liberty.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>
<uninstall modify='false'>
<offering id='com.ibm.websphere.liberty.IBMJAVA.v80'
 profile='WebSphere Liberty V8.5'/>
</uninstall>
<profile id='WebSphere Liberty V8.5'
 installLocation='C:\Program Files\IBM\WebSphere\Liberty'>
</profile>
</agent-input>
```

## Using the sample response files

You can edit and use sample response file for installing, modifying, or uninstalling Liberty offerings silently.

## Procedure

- “Sample response file: Installing IBM WebSphere Application Server Liberty Core” on page 763
- “Sample response file: Modifying IBM WebSphere Application Server Liberty Core” on page 767

- “Sample response file: Uninstalling IBM WebSphere Application Server Liberty Core” on page 772
- “Sample response file: Installing IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty” on page 775
- “Sample response file: Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty” on page 780

### Sample response file: Installing IBM WebSphere Application Server Liberty Core:

You can edit and use this example of a response file for installing IBM WebSphere Application Server Liberty Core.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
Question 3. How do I store and pass credentials to repositories that
require authentication?
Answer 3. There are two methods for storing authentication credentials
for Installation Manager depending on the version being used,
either key ring files or storage files.
#
Versions of Installation Manager before 1.6.2 use a key ring file to store
encrypted credentials for authenticating with repositories. Follow this
two-step process for creating and using a key ring file with Installation Manager.
#
First, create a key ring file with your credentials by starting
Installation Manager from the command line under eclipse subdirectory
with the keyring parameter.
Use the optional password parameter to password protect your file.
#
```

```

Windows = IBMIM.exe -keyring <path and file name> -password <password>
Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
-password <password>
#
Installation Manager will start in graphical mode. Verify that the
repositories to which you need to authenticate are included in the
preferences, File / Preferences / Repositories. If they are not
listed, then click Add Repositories to add the URL or UNC path.
Installation Manager will prompt for your credentials. If the repository
is already in the list, then any attempt to access the repository location,
such as clicking the Test Connections button, will also prompt for your
credentials. Enter the correct credential and check the Save password
checkbox. The credentials are saved to the key ring file you specified.
#
Second, when you start a silent installation, run imcl under eclipse/tools
subdirectory, and provide Installation Manager with the location of the key
ring file and the password if the file is protected. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
#
Versions of Installation Manager at 1.6.2 or higher use storage files
to store encrypted credentials. Complete the following steps to create
master password and storage files to use with Installation Manager.
#
First, if you do not have a master password file already, create a text file
that contains a master passphrase. An example of a passphrase is:
"This text is the passphrase for a master password file."
#
Next, run imutilsc under the eclipse/tools subdirectory with the following
options to create and store user credentials in a storage file.
-secureStorageFile <path and file name of storage file>
-masterPasswordFile <path and file name of master password file>
-url <repository address> or -passportAdvantage <PPA repository address>
-userName <user name>
-userPassword <password for user>
#
Example of a command to create a storage file by operating system
Windows = imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile C:\IBM\credential.store
-masterPasswordFile C:\IBM\master_password_file.txt
Linux, UNIX, IBM z/OS, and the OS X operating system =
./imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile /home/IBM/credential.store
-masterPasswordFile /home/user/IBM/master_password_file.txt
#
Usage hints:
* Do not use both the -url and -passportAdvantage options in the same command.
* Enclose file paths that include spaces with double quotation marks.
* If you use the IBM service repositories, you can specify the value:
http://www.ibm.com/software/repositorymanager/entitled/repository.xml
for the -url option which is a generic service repository for IBM packages.
* Repeat steps to store credentials for multiple users in one file.
* Repeat steps to store credentials for multiple repositories in one file.
#
Afterwards, when you start a silent installation, run imcl under the eclipse/tools
subdirectory, and provide Installation Manager with the location of the storage
file. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
#
-->
<!-- ##### Agent Input #####
#
Note that the acceptLicense attribute has been deprecated.
Use the -acceptLicense command line option to accept license agreements.
#
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the installation finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:

```

```

true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
 <!-- ##### IBM WebSphere Live Update Repositories #####
 # These repositories contain WebSphere Application Server Liberty offerings,
 # and updates for those offerings
 #
 # To use the secure repository (https), you must have an IBM ID,
 # which can be obtained by registering at: http://www.ibm.com/account
 # or your Passport Advantage account.
 #
 # And, you must use a key ring file with your response file.
 ##### -->
 <repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85"/>
 <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

 <!-- ##### Custom Repositories #####
 # Uncomment and update the repository location key below
 # to specify URLs or UNC paths to any intranet repositories
 # and directory paths to local repositories to use.
 ##### -->
 <!-- <repository location="https:\w3.mycompany.com\repositories\" /> -->
 <!-- <repository location="/home/user/repositories/websphere/" /> -->

 <!-- ##### Local Repositories #####
 # Uncomment and update the following line when using a local
 # repository located on your own machine to install a
 # WebSphere Application Server Liberty offering.
 ##### -->
 <!-- <repository location="insert the full directory path inside single quotes" /> -->
</server>

<!-- ##### Install Packages #####
#
Install Command
#
Use the install command to inform Installation Manager of the
installation packages to install.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be installed. The offering listed must be present in
at least one of the repositories listed earlier. The example
command below contains the offering ID for the Core
edition of WebSphere Application Server Liberty.
#
The version attribute is optional. If a version number is provided,
then the offering will be installed at the version level specified
as long as it is available in the repositories. If the version
attribute is not provided, then the default behavior is to install
the latest version available in the repositories. The version number
can be found in the repository.xml file in the repositories.
For example, <offering ... version="8.5.5000.20130326_0211">.
#
The profile attribute is required and typically is unique to the
offering. If modifying or updating an existing installation, the
profile attribute must match the profile ID of the targeted installation
of WebSphere Application Server Liberty.
#

```

```

The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer
#
The features embeddablecontainer is a subfeature of liberty.
#
You can use these functions to add or remove feature embeddablecontainer later.
#
The installFixes attribute indicates whether fixes available in
repositories are installed with the product. By default, all
available fixes will be installed with the offering.
#
Valid values for installFixes:
none = do not install available fixes with the offering.
recommended = installs all available recommended fixes with the offering.
all = installs all available fixes with the offering.
#
Interim fixes for offerings also can be installed while they
are being installed by including the offering ID for the interim
fix and specifying the profile ID.
#
Installation Manager supports installing multiple offerings at once.
Additional offerings can be included in the install command,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=/com.ibm.websphere.wlp.core.doc%2Fae%2Ftwlp_ins_installation_dist_silent.html
#
-->

<install modify="false">
<offering id="com.ibm.websphere.liberty.v85" profile="WebSphere Liberty V8.5" features="liberty,embeddablecontainer" installFixes="none" />
</install>
<profile id="WebSphere Liberty V8.5" installLocation="C:\Program Files\IBM\WebSphere\Liberty">
<data key="eclipseLocation" value="C:\Program Files\IBM\WebSphere\Liberty" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#

```

```

First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false' />
-->

</agent-input>

```

## Sample response file: Modifying IBM WebSphere Application Server Liberty Core:

You can edit and use this example of a response file for modifying IBM WebSphere Application Server Liberty Core.

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.

```

```

#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse/tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
Question 3. How do I store and pass credentials to repositories that
require authentication?
Answer 3. There are two methods for storing authentication credentials
for Installation Manager depending on the version being used,
either key ring files or storage files.
#
Versions of Installation Manger before 1.6.2 use a key ring file to store
encrypted credentials for authenticating with repositories. Follow this
two-step process for creating and using a key ring file with Installation Manager.
#
First, create a key ring file with your credentials by starting
Installation Manager from the command line under eclipse subdirectory
with the keyring parameter.
Use the optional password parameter to password protect your file.
#
Windows = IBMIM.exe -keyring <path and file name> -password <password>
Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
-password <password>
#
Installation Manager will start in graphical mode. Verify that the
repositories to which you need to authenticate are included in the
preferences, File / Preferences / Repositories. If they are not
listed, then click Add Repositories to add the URL or UNC path.
Installation Manager will prompt for your credentials. If the repository
is already in the list, then any attempt to access the repository location,
such as clicking the Test Connections button, will also prompt for your
credentials. Enter the correct credential and check the Save password
checkbox. The credentials are saved to the key ring file you specified.
#
Second, when you start a silent installation, run imcl under eclipse/tools
subdirectory, and provide Installation Manager with the location of the key
ring file and the password if the file is protected. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
#
Versions of Installation Manager at 1.6.2 or higher use storage files
to store encrypted credentials. Complete the following steps to create
master password and storage files to use with Installation Manager.
#
First, if you do not have a master password file already, create a text file
that contains a master passphrase. An example of a passphrase is:
"This text is the passphrase for a master password file."
#
Next, run imutilsc under the eclipse/tools subdirectory with the following
options to create and store user credentials in a storage file.
-secureStorageFile <path and file name of storage file>
-masterPasswordFile <path and file name of master password file>
-url <repository address> or -passportAdvantage <PPA repository address>
-userName <user name>
-userPassword <password for user>
#
Example of a command to create a storage file by operating system
Windows = imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile C:\IBM\credential.store

```



```

-masterPasswordFile C:\IBM\master_password_file.txt
Linux, UNIX, IBM z/OS, and the OS X operating system =
./imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile /home/IBM/credential.store
-masterPasswordFile /home/user/IBM/master_password_file.txt
#
Usage hints:
* Do not use both the -url and -passportAdvantage options in the same command.
* Enclose file paths that include spaces with double quotation marks.
* If you use the IBM service repositories, you can specify the value:
http://www.ibm.com/software/repositorymanager/entitled/repository.xml
for the -url option which is a generic service repository for IBM packages.
* Repeat steps to store credentials for multiple users in one file.
* Repeat steps to store credentials for multiple repositories in one file.
#
Afterwards, when you start a silent installation, run imcl under the eclipse/tools
subdirectory, and provide Installation Manager with the location of the storage
file. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
#
-->

<!-- ##### Agent Input #####
#
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the installation finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
 <!-- ##### IBM WebSphere Live Update Repositories #####
 # These repositories contain WebSphere Application Server Liberty offerings,
 # and updates for those offerings
 #
 # To use the secure repository (https), you must have an IBM ID,
 # which can be obtained by registering at: http://www.ibm.com/account
 # or your Passport Advantage account.
 #
 # And, you must use a key ring file with your response file.
 ##### -->
 <repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85"/>
 <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

 <!-- ##### Custom Repositories #####
 # Uncomment and update the repository location key below
 # to specify URLs or UNC paths to any intranet repositories
 # and directory paths to local repositories to use.
 ##### -->
 <!-- <repository location="https://w3.mycompany.com/repositories/" /> -->
 <!-- <repository location="/home/user/repositories/websphere/" /> -->

```

```

<!-- ##### Local Repositories #####
Uncomment and update the following line when using a local
repository located on your own machine to install a
WebSphere Application Server Liberty offering.
-->
<!-- <repository location='insert the full directory path inside single quotes' /> -->
</server>

<!-- ##### Modify Packages #####
#
Install and Uninstall Commands
#
Use the install and uninstall commands to inform Installation Manager
of the installation packages to install or uninstall.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be installed. The offering listed must be present in
at least one of the repositories listed earlier. The example
command below contains the offering ID for the Core
edition of WebSphere Application Server Liberty.
#
The version attribute is optional. If a version number is provided,
then the offering will be installed or uninstalled at the version level
specified as long as it is available in the repositories. If the version
attribute is not provided, then the default behavior is to install or
uninstall the latest version available in the repositories. The version
number can be found in the repository.xml file in the repositories.
For example, <offering ... version='8.5.5000.20130328_1111'>.
#
The profile attribute is required and typically is unique to the
offering. If modifying or updating an existing installation, the
profile attribute must match the profile ID of the targeted installation
of WebSphere Application Server Liberty.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer
#
The features embeddablecontainer is a subfeature of liberty.
#
You can use these functions to add or remove feature embeddablecontainer
later.
#
In the example that follows, the feature embeddablecontainer is
being added and no feature are being removed from the specified offering.
#
The installFixes attribute indicates whether fixes available in
repositories are installed with the product. By default, all
available fixes will be installed with the offering.
#
Valid values for installFixes:
none = do not install available fixes with the offering.
recommended = installs all available recommended fixes with the offering.
all = installs all available fixes with the offering.
#
Installation Manager supports modifying multiple offerings at once.
Additional offerings can be included in the install and uninstall commands,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on

```

```

a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Fwlp_ins_installation_dist_silent.html
#
-->

<install modify="true">
<offering id="com.ibm.websphere.liberty.v85" profile="WebSphere Liberty V8.5" features="embeddablecontainer" />
</install>
<profile id="WebSphere Liberty V8.5" installLocation="C:\Program Files\IBM\WebSphere\Liberty">
<data key="eclipseLocation" value="C:\Program Files\IBM\WebSphere\Liberty" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />

```

```

<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false'/>
-->

</agent-input>

```

## Sample response file: Uninstalling IBM WebSphere Application Server Liberty Core:

You can edit and use this example of a response file for uninstalling IBM WebSphere Application Server Liberty Core.

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
-->

<!-- ##### Agent Input #####
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the uninstall finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the

```

```

response file persist in Installation Manager.
#
-->
<agent-input clean="true" temporary="true">
<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
 <!-- ##### IBM WebSphere Live Update Repositories #####
 # These repositories contain WebSphere Application Server Liberty offerings,
 # and updates for those offerings
 #
 # To use the secure repository (https), you must have an IBM ID,
 # which can be obtained by registering at: http://www.ibm.com/account
 # or your Passport Advantage account.
 #
 # And, you must use a key ring file with your response file.
 ##### -->
 <!--repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85"/-->
 <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

 <!-- ##### Custom Repositories #####
 # Uncomment and update the repository location key below
 # to specify URLs or UNC paths to any intranet repositories
 # and directory paths to local repositories to use.
 ##### -->
 <!-- <repository location='https://w3.mycompany.com/repositories/' /> -->
 <!-- <repository location='/home/user/repositories/websphere/' /> -->

 <!-- ##### Local Repositories #####
 # Uncomment and update the following line when using a local
 # repository located on your own machine to install a
 # WebSphere Application Server Liberty offering.
 ##### -->
 <!-- <repository location='insert the full directory path inside single quotes/' /> -->
</server>

<!-- ##### Uninstall Packages #####
#
Uninstall Command
#
Use the uninstall command to inform Installation Manager of the
installation packages to uninstall.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be uninstalled. The example command below contains the
offering ID for WebSphere Application Server Liberty Core edition.
#
The version attribute is optional. If a version number is provided,
then the offering will be uninstalled at the version level specified
If the version attribute is not provided, then the default behavior is
to uninstall the latest version. The version number can be found in
the repository.xml file in the repositories.
For example, <offering ... version='8.5.5000.20130326_0211'>.
#
The profile attribute is required and must match the package group
name for the offering to be uninstalled.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer

```

```

#
The features embeddablecontainer is a subfeature of liberty.
#
Installation Manager supports uninstalling multiple offerings at once.
Additional offerings can be included in the uninstall command,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Ftwlp_ins_installation_dist_silent.html
#
-->
<uninstall modify="false">
<offering id="com.ibm.websphere.liberty.v85" profile="WebSphere Liberty V8.5" features="liberty,embeddablecontainer" />
</uninstall>
<profile id="WebSphere Liberty V8.5" installLocation="C:\Program Files\IBM\WebSphere\Liberty">
<data key="eclipseLocation" value="C:\Program Files\IBM\WebSphere\Liberty" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical

```

```

interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->
<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45'/>
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false'/>
-->

</agent-input>

```

## Sample response file: Installing IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty:

You can edit and use this example of a response file for installing IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty.

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress

```

```

input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
Question 3. How do I store and pass credentials to repositories that
require authentication?
Answer 3. There are two methods for storing authentication credentials
for Installation Manager depending on the version being used,
either key ring files or storage files.
#
Versions of Installation Manager before 1.6.2 use a key ring file to store
encrypted credentials for authenticating with repositories. Follow this
two-step process for creating and using a key ring file with Installation Manager.
#
First, create a key ring file with your credentials by starting
Installation Manager from the command line under eclipse subdirectory
with the keyring parameter.
Use the optional password parameter to password protect your file.
#
Windows = IBMIM.exe -keyring <path and file name> -password <password>
Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
-password <password>
#
Installation Manager will start in graphical mode. Verify that the
repositories to which you need to authenticate are included in the
preferences, File / Preferences / Repositories. If they are not
listed, then click Add Repositories to add the URL or UNC path.
Installation Manager will prompt for your credentials. If the repository
is already in the list, then any attempt to access the repository location,
such as clicking the Test Connections button, will also prompt for your
credentials. Enter the correct credential and check the Save password
checkbox. The credentials are saved to the key ring file you specified.
#
Second, when you start a silent installation, run imcl under eclipse/tools
subdirectory, and provide Installation Manager with the location of the key
ring file and the password if the file is protected. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
#
Versions of Installation Manager at 1.6.2 or higher use storage files
to store encrypted credentials. Complete the following steps to create
master password and storage files to use with Installation Manager.
#
First, if you do not have a master password file already, create a text file
that contains a master passphrase. An example of a passphrase is:
"This text is the passphrase for a master password file."
#
Next, run imutilsc under the eclipse/tools subdirectory with the following
options to create and store user credentials in a storage file.
-secureStorageFile <path and file name of storage file>
-masterPasswordFile <path and file name of master password file>
-url <repository address> or -passportAdvantage <PPA repository address>
-userName <user name>
-userPassword <password for user>
#
Example of a command to create a storage file by operating system
Windows = imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile C:\IBM\credential.store
-masterPasswordFile C:\IBM\master_password_file.txt
Linux, UNIX, IBM z/OS, and the OS X operating system =
./imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile /home/IBM/credential.store
-masterPasswordFile /home/user/IBM/master_password_file.txt
#
Usage hints:
* Do not use both the -url and -passportAdvantage options in the same command.
* Enclose file paths that include spaces with double quotation marks.
* If you use the IBM service repositories, you can specify the value:
http://www.ibm.com/software/repositorymanager/entitled/repository.xml
for the -url option which is a generic service repository for IBM packages.
* Repeat steps to store credentials for multiple users in one file.
* Repeat steps to store credentials for multiple repositories in one file.
#
Afterwards, when you start a silent installation, run imcl under the eclipse/tools
subdirectory, and provide Installation Manager with the location of the storage
file. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>

```



```

-masterPasswordFile <path and name of master password file>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
#
-->

<!-- ##### Agent Input #####
#
Note that the acceptLicense attribute has been deprecated.
Use the -acceptLicense command line option to accept license agreements.
#
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the installation finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>

<!-- ##### IBM WebSphere Live Update Repositories #####
These repositories contain IBMJAVA for WebSphere Application Server
Liberty offerings, and updates for those offerings
#
To use the secure repository (https), you must have an IBM ID,
which can be obtained by registering at: http://www.ibm.com/account
or your Passport Advantage account.
#
And, you must use a key ring file with your response file.
-->

<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v70"/>

<!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

<!-- ##### Custom Repositories #####
Uncomment and update the repository location key below
to specify URLs or UNC paths to any intranet repositories
and directory paths to local repositories to use.
-->
<!-- <repository location="https://w3.mycompany.com/repositories"/> -->
<!-- <repository location="/home/user/repositories/websphere"/> -->

<!-- ##### Local Repositories #####
Uncomment and update the following line when using a local
repository located on your own machine to install a
IBMJAVA for WebSphere Application Server Liberty offering.
-->
<!-- <repository location='insert the full directory path inside single quotes' /> -->

</server>

<!-- ##### Install Packages #####
#
Install Command
#
Use the install command to inform Installation Manager of the
installation packages to install.

```

```

#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be installed. The offering listed must be present in
at least one of the repositories listed earlier. The example
command below contains the offering ID for the IBM WebSphere SDK Java
Technology Edition (Optional) .
#
The version attribute is optional. If a version number is provided,
then the offering will be installed at the version level specified
as long as it is available in the repositories. If the version
attribute is not provided, then the default behavior is to install
the latest version available in the repositories. The version number
can be found in the repository.xml file in the repositories.
For example, <offering ... version='7.0.4000.20130328_1111'>.
#
The profile attribute is required and typically is unique to the
offering. If modifying or updating an existing installation, the
profile attribute must match the profile ID of the targeted installation
of IBMJAVA for WebSphere Application Server Liberty.
#
IBM WebSphere SDK Java Technology Edition (Optional) 7.0.4.0 is an
extension offering which requires a base WebSphere Application
Server / Client product to be installed first The profile id for
IBM WebSphere SDK Java Technology Edition (Optional) 7.0.4.0 must be
the same as the base product.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The installFixes attribute indicates whether fixes available in
repositories are installed with the product. By default, all
available fixes will be installed with the offering.
#
Valid values for installFixes:
none = do not install available fixes with the offering.
recommended = installs all available recommended fixes with the offering.
all = installs all available fixes with the offering.
#
Interim fixes for offerings also can be installed while they
are being installed by including the offering ID for the interim
fix and specifying the profile ID. A commented out example is
provided in the install command below.
#
Installation Manager supports installing multiple offerings at once.
Additional offerings can be included in the install command,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
Include data keys for product specific profile properties.
For instance, Installing IBMJAVA for Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Ftwlp_ins_installation_dist_silent.html
#
-->

<install modify="false">
<offering id="com.ibm.websphere.liberty.IBMJAVA.v70" profile="WebSphere Liberty V8.5" features="com.ibm.sdk.7" installFixes="none" />
</install>
<profile id="WebSphere Liberty V8.5" installLocation="C:\Program Files\IBM\WebSphere\Liberty">
<data key="eclipseLocation" value="C:\Program Files\IBM\WebSphere\Liberty" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

```

```

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false' />
<preference name='com.ibm.cic.common.sharedUI.showErrorLog' value='true' />
<preference name='com.ibm.cic.common.sharedUI.showWarningLog' value='true' />
<preference name='com.ibm.cic.common.sharedUI.showNoteLog' value='true' />
-->

</agent-input>

```

## Sample response file: Uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty:

You can edit and use this example of a response file for uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
-->

<!-- ##### Agent Input #####
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the uninstall finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
```

```

Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->
<!-- ##### Uninstall Packages ##### -->
#
Uninstall Command
#
Use the uninstall command to inform Installation Manager of the
installation packages to uninstall.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be uninstalled. The example command below contains the
offering ID for IBM WebSphere SDK Java Technology Edition (Optional) 7.0.4.0.
#
The version attribute is optional. If a version number is provided,
then the offering will be uninstalled at the version level specified
If the version attribute is not provided, then the default behavior is
to uninstall the latest version. The version number can be found in
the repository.xml file in the repositories.
For example, <offering ... version='7.0.4000.20130328_1111'>.
#
The profile attribute is required and must match the package group
name for the offering to be uninstalled.
#
The features attribute is optional. If there is no feature attribute,
then all features are uninstalled. If features are specified, then
only those features will be uninstalled.
Features must be comma delimited without spaces.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
IBM WebSphere SDK Java Technology Edition (Optional) 7.0.4.0 as the installLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Fwlp_ins_installation_dist_silent.html
#
-->
<uninstall modify="false">
<offering id="com.ibm.websphere.liberty.IBMJAVA.v70" profile="WebSphere Liberty V8.5" features="com.ibm.sdk.7" />
</uninstall>
<profile id="WebSphere Liberty V8.5" installLocation="C:\Program Files\IBM\WebSphere\Liberty">
<data key="eclipseLocation" value="C:\Program Files\IBM\WebSphere\Liberty" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager

```

```

Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false' />
<preference name='com.ibm.cic.common.sharedUI.showErrorLog' value='true' />
<preference name='com.ibm.cic.common.sharedUI.showWarningLog' value='true' />
<preference name='com.ibm.cic.common.sharedUI.showNoteLog' value='true' />
-->

</agent-input>

```

## Creating custom installation repositories with IBM Packaging Utility

IBM WebSphere Application Server Liberty uses IBM Installation Manager for installation and lifecycle management. Installation Manager accesses source repositories that contain the content for a software product installation. Repositories are available on product media, in IBM-hosted web-based repositories, and from Passport Advantage. IBM Packaging Utility can help you create and customize enterprise repositories that contain the correct combination of products and maintenance levels needed for all aspects of your business.

## About this task

You use Installation Manager to connect to an Installation Manager repository (or set of repositories) to find products and service updates that are available to you for installation. An Installation Manager repository is simply a tree-structured file folder that includes product payload and metadata. You can install the software products that you need directly from an IBM web-based service repository or download and unpack compressed files from Passport Advantage and install the products from the resulting unpacked file folders. The result of unpacking the files is also considered to be an Installation Manager repository. Like any Installation Manager repository, these unpacked files can be hosted on an internal HTTP server, FTP server, or network mount in order to make them available to the organization.

Packaging Utility is a companion tool for Installation Manager with which you can create and manage custom Installation Manager repositories for your organization. You can copy multiple packages, maintenance levels, and fixes into a single repository. Packaging Utility copies from source repositories to your target custom repositories. Source repositories can include any accessible Installation Manager repository, including IBM web-hosted product repositories and unzipped Passport Advantage downloads.

For more information on Packaging Utility, go to the IBM Packaging Utility Information Center.

## Procedure

Use Packaging Utility to create custom or "enterprise" Installation Manager repositories that contain specific products and maintenance levels that fit the needs of your business.

As an administrator, you can control the content of your enterprise repository, which then can serve as the central repository to which your organization connects in order to perform product installations and updates.

Packaging Utility essentially copies from a set of source Installation Manager repositories to a target repository and eliminates duplicate artifacts, helping to keep the repository size as small as possible. You can also delete (or "prune") a repository, removing maintenance levels or products that are not needed. You can download the latest version of Packaging Utility from the IBM Support Portal.

Like Installation Manager, Packaging Utility has GUI and command-line interfaces. You must specify repository URLs for Installation Manager repositories that contain the offerings that you wish to copy. Installation Manager repository URLs follow this pattern:

`http://www.ibm.com/software/repositorymanager/offering_name`

**Note:** This location does not contain a web page that you can access using a web browser.

For example, WebSphere Application Server Liberty product repositories are located at the following URLs:

- WebSphere Application Server Liberty Core Version 8.5  
`http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85`
- WebSphere Application Server Liberty Core Trial Version 8.5  
`http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.TRIAL.v85`

See Online product repositories for WebSphere Application Server offerings for additional product repositories.

The target repository that you create with Packaging Utility will always support a full installation; therefore, you cannot use Packaging Utility to create a repository that is only a copy of a fix pack. You can, however, create a repository that contains the minimum content to support direct installation to a fix-pack level. Consider the following two examples that use the Packaging Utility command-line interface (**PUCL.exe**) that is available in the Packaging Utility installation folder.

- **Example 1**

**Note:** Note that you must read the license agreement that you can find with the product files and then signify your acceptance of the license agreement by specifying **-acceptLicense** in the command as shown in the following example.

```

PUCL copy com.ibm.websphere.liberty.v85
-repositories
 http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-target D:\LIB_CORE_version
-prompt
-showProgress
-acceptLicense

```

Since no version number is specified with the offering name, this command will create a new repository that supports direct installation to the latest fix-pack level for WebSphere Application Server Liberty Version 8.5. This new repository does not support the installation of Version 8.5.5.0, but it does support the update from an existing Version 8.5.5.0 installation to the latest version.

- **Example 2**

```

PUCL copy com.ibm.websphere.liberty.v85_8.5.5.0.20110503_0200
-repositories
 http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-target D:\LIB_CORE
-prompt
-showProgress
-acceptLicense

```

```

PUCL copy com.ibm.websphere.liberty.v85_8.5.5.1.20110829_1838
-repositories
 http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-target D:\LIB_CORE
-prompt
-showProgress
-acceptLicense

```

The first command creates a target repository with WebSphere Application Server Liberty Version 8.5.5.0. The second command adds the Version 8.5.5.1 fix pack to the same repository. You can now use this resulting repository to install Version 8.5.5.0, install Version 8.5.5.1, or update from Version 8.5.5.0 to Version 8.5.5.1.

**Tip:** With some offerings, such as WebSphere SDK Java Technology Edition Version 7.0 for Liberty, you can use Packaging Utility with the **-platform** option (sometimes called "platform slicing") to create a repository that is scoped to the platforms and architectures that are used by your organization. This feature is available in command-line mode by specifying the **-platform** option with the `os` and `arch` arguments as shown in the following example:

```

PUCL copy com.ibm.websphere.liberty.IBMJAVA.v70
-repositories http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.IBMJAVA.v70
-platform os=linux,arch=ppc64
-target D:\LIB_CORE
-prompt
-showProgress
-acceptLicense</p>

```

Your repository can be scoped for platforms other than the one on which it is created or stored. For example, you can run Packaging Utility on a Windows system to create a repository with the content needed to install on a Linux system. During installation on Linux, you point Installation Manager to your custom repository.

The following table lists valid combinations for creating a local WebSphere SDK Java Technology Edition Version 7.0 for Liberty offering repository that is sliced by operating system and architecture.



Table 62. Valid combinations for creating a local WebSphere SDK Java Technology Edition Version 7.0 for Liberty offering repository using the Packaging Utility

Platform	Options	Resulting Repository
Windows	os=win32,arch=x86	32-bit repository for 32-bit Windows OS and 64-bit Windows
	os=win32.arch=x86_64	64-bit repository for 64-bit Windows
Linux Intel	os=linux,arch=x86	32-bit repository for 32-bit Linux Intel and 64-bit Linux Intel
	os=linux.arch=x86_64	64-bit repository for 64-bit Linux Intel
Linux Power®	os=linux,arch=ppc	32-bit repository for 32-bit Linux Power and 64-bit Linux Power
	os=linux.arch=ppc64	64-bit repository for 64-bit Linux Power
zLinux	os=linux,arch=s390	32-bit repository for 32-bit zLinux and 64-bit zLinux
	os=linux.arch=s390x	64-bit repository for 64-bit zLinux
AIX®	os=aix,arch=ppc	32-bit repository for 32-bit AIX and 64-bit AIX
	os=aix.arch=ppc64	64-bit repository for 64-bit AIX
Solaris Sparc	os=solaris,arch=sparc	32-bit repository for 32-bit Solaris Sparc and 64-bit Solaris Sparc
	os=solaris,arch=sparc64	64-bit repository for 64-bit Solaris Sparc
Solaris Intel	os=solaris,arch=x86_64	64-bit repository for 64-bit Solaris Intel
HP-UX Itanium	os=hpux,arch=ia64	64-bit repository for 64-bit HP-UX Itanium
z/OS	os=zos,arch=s390x	64-bit repository for z/OS

For more information on platform slicing, go to the IBM Packaging Utility Information Center.

## Using the launchpad to start Liberty installations

The launchpad console is the starting point for installing IBM WebSphere Application Server Liberty Core.

### Before you begin

- The launchpad is a web application. Before using the launchpad, you must have a supported web browser. The launchpad supports the following browsers:
  - **AIX** **HP-UX** **Linux** **Solaris** Mozilla Firefox Version 3.5 or later
  - **Windows** Internet Explorer Version 6.0 Service Pack 2 or later
- Install a supported web browser if one is not installed.
  - **AIX** **HP-UX** **Linux** **Solaris** Install a browser such as Mozilla Firefox. Download Firefox from the following location: <http://www.mozilla.org/products/firefox/>.
  - **Windows** Install a browser for the Windows operating system.
    - Download Internet Explorer from the following location: <http://www.microsoft.com/windows/ie/default.msp>
    - Download Mozilla Firefox from the following location: <http://www.mozilla.org/products/firefox/>.

- **AIX** **HP-UX** **Linux** **Solaris** You must install the Bash shell package to use the launchpad application. Although the Bash shell must be installed, the Bash shell does not need to be used to run the **launchpad.sh** command. If you attempt to run the launchpad application from a DVD on the HP-UX, Linux, or Solaris operating systems without the Bash shell installed, the launchpad fails with an error message indicating that the Bash interpreter is not found. If you attempt to run the launchpad from any image on AIX, the launchpad fails with an error message indicating that the current browser is not supported. The Bash package for the AIX operating system is included in the IBM AIX Toolbox.

#### Examples of what the launchpad can do:

- The launchpad does support installing Installation Manager in admin or non-admin (user) mode.
- The launchpad does support updating Installation Manager from an earlier version in admin or non-admin (user) mode.

#### Examples of what the launchpad cannot do:

- The launchpad does not support installing in group mode.
- The launchpad does not support installing or updating with a custom application data location.

**Restriction:** You cannot run the launchpad remotely to install a product. Only local use of the launchpad is supported.

#### About this task

The launchpad identifies components on the product disk or image that you can install (launch).

The launchpad is a single point of reference for installing the application server environment. If you click a link that points to a product repository on another disk or image, you are prompted to insert that disk or browse to that image.

#### Procedure

1. Start the launchpad.

The launchpad program is available in the root directory of the product disk. You can start the launchpad manually using a fully qualified command instead of changing directories to the disk and running the command locally from the root directory:

- **AIX** **HP-UX** **Linux** **Solaris** Mount the disk drive if necessary.
- Open a shell window and issue a fully qualified **launchpad** command to start the launchpad.

**Tip:** **Windows** If you need to navigate using the keyboard, use Mozilla Firefox as your web browser and start the launchpad with the following command:

```
launchpad_all1y.exe
```

**Note:** **Windows** Some Windows operating systems such as Windows 2003, Windows Vista, Windows Server 2008, and Windows 7 have implemented a more restrictive security policy that denies access to trusted files by non-trusted files or applications. When the launchpad application is run as a non-trusted program, you will receive JavaScript “Access is denied” errors that subsequently cause the application to hang. Because downloaded images are automatically blocked, unblock the files so that the launchpad can successfully access the files. Before you extract the image, right-click the image file and select **Properties** to open the Properties panel and locate the security section and click the **Unblock** button. You can now extract the image and run the launchpad application.

The launchpad opens in the language of the locale setting of the machine.

2. Use the launchpad to perform the following Liberty Core related tasks.
  - View the Welcome page, and access links to the IBM WebSphere Application Server Information Center and the IBM Education Assistant.

- Launch IBM Installation Manager installation, and access the IBM Installation Manager Information Center.
- Launch IBM Packaging Utility installation, and access the Packaging Utility information in the IBM Installation Manager Information Center.
- Launch Installation Manager to install WebSphere Application Server Liberty Core, and access the installation instructions in the information center.
- Launch Installation Manager to install IBM HTTP Server, and access the installation instructions in the information center.
- Launch Installation Manager to install Web Server Plug-ins for IBM WebSphere Application Server, and access the installation instructions in the information center.
- Access the latest version of IBM Support Assistant.
- Launch Installation Manager to install IBM WebSphere SDK Java Technology Edition Version 7.0 for Liberty, and access the installation instructions in the information center.
- Launch Installation Manager to install the IBM WebSphere Application Server Web 2.0 and Mobile Toolkit, and access the installation instructions in the information center.

## Results

This procedure results in using the launchpad to start the installation and to access information through a browser.

### Troubleshooting

If you can start the launchpad but clicking a link does not resolve to a page in the launchpad, you might have the wrong media in the disk drive. Check the validity of the media.

Use the following procedure to correct any error that is preventing the launchpad from displaying. Then, try to start the launchpad again:

1. If the product disk is no longer accessible, insert the disk.
2. AIX HP-UX Linux Solaris Mount the drive as necessary on platforms such as AIX or Linux.
3. Enable the JavaScript function in your browser.
 

Mozilla Firefox: Click **Tools > Options > Content**:

  - Select **Enable Java**.
  - Select **Enable JavaScript**.
  - Click **Advanced** and allow scripts to ... (Select all boxes.)

Windows Internet Explorer: Click **Tools > Internet Options > Security > Custom Level for Internet > Scripting > Active scripting > Enable**.
4. Restart the launchpad by issuing the **launchpad** command.

If the launchpad links still do not work after following this procedure, launch Installation Manager and point it at the appropriate repositories to install the offerings.

### Distributed operating systems

## What to do next

Go to “Installing and uninstalling Liberty on distributed operating systems” on page 691 to continue installing your application serving environment.

## Installing and uninstalling Liberty on IBM i operating systems

IBM Installation Manager is a common installer for many IBM software products. You can use Installation Manager to install and manage the product lifecycle of WebSphere Application Server Liberty Core.

### Before you begin

**Note:** 8.5.5.11 Support for using Java SE 6 with WebSphere Liberty ends in September 2017. After the end of support, the Liberty kernel will be recompiled and can no longer run with Java SE 6. If you continue to use Java SE 6 on earlier fix packs after the end of support date, you could expose your environment to security risks.

Java SE 8 is the recommended Java SDK because it provides the latest features and security updates.

Installation Manager is a single installation program that can use remote or local software repositories to install, modify, or update WebSphere Application Server Liberty Core. It determines available packages - including products, fix packs, interim fixes, and so on - checks prerequisites and interdependencies, and installs the selected packages. You also use Installation Manager to uninstall the packages that it installed.

**Restriction:** The Installation Manager GUI is not available on IBM i; all interaction with Installation Manager on IBM i is done through the command line or response files.

**Overview of IBM Installation Manager:** IBM Installation Manager is a general-purpose software installation and update tool that runs on a range of computer systems. Installation Manager can be invoked through a command-line interface. You can also create response files in XML and use them to direct the performance of Installation Manager tasks in silent mode.

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

**Packages and package groups:** Each software product that can be installed with Installation Manager is referred to as a package. An installed package has a product level and an installation location. A package group consists of all of the products that are installed at a single location.

**How many Installation Managers do you need:** You only need to run Installation Manager on those systems on which you install or update product code. You normally need only one Installation Manager on a system because one Installation Manager can keep track of any number of product installations.

**Creating an Installation Manager:** When the installation kit is available on your system, you can create an Installation Manager. An Installation Manager consists of a set of binaries that are copied from the installation kit and a set of runtime data that describe the products that have been installed by this particular Installation Manager. Before creating an Installation Manager, you must decide in which mode the Installation Manager will run as well as where the binaries and runtime data - called agent data or appdata - will reside. Then, you issue the Installation Manager installation command from the appropriate user ID to create the Installation Manager.

**Accessing product repositories:** All software materials that will be installed with IBM Installation Manager are stored in repositories. Each repository contains program objects and metadata for one or more packages - that is, software products at a particular level. Repositories can also contain product maintenance, such as fix packs and interim fixes. Whenever you install a new product, you can choose from any of the available product levels in any accessible repository.

**Installing the product:** After you have created an Installation Manager and have access to all necessary product repositories, you can use Installation Manager command-line commands or response files to perform the actual product installations. When you install a product, you provide the package name, optionally the product level to be installed, the product location, and any other optional properties. For

example, some products have optional features that you can select at installation time or a list of optional supported language packs from which you can select.

**Working with installed products:** You can use Installation Manager commands to list installed products and product levels. You can also obtain this information for installed copies of WebSphere Application Server LibertyCore by issuing the **versionInfo** command from the product file system. You can use Installation Manager commands or response files to install a new product level, roll back to a previous level, or modify the product by adding or removing optional features or language packs.

#### Notes:

- You must have Java SE 6 32 bit (option 11 of the IBM Developer Kit for Java) installed on your IBM i system before installing WebSphere Application Server LibertyCore. For more information, read IBM i prerequisites.
- Do not transfer the content of a repository in non-binary mode and do not convert any content on extraction.
- When you try to install IBM Installation Manager locally from the WebSphere Application Server LibertyCore media on an IBM i operating system, the following error message might be displayed:  
The Installc executable launcher was unable to locate its companion shared library.

This error occurs because all directory and files names contained by the media are displayed in uppercase. To resolve this issue, enable the handling of mixed case on your IBM i operating system using the following command:

```
CHGOPTA EXTMEDFMT(*YES)
```

- **8.5.5.1** If you are installing WebSphere Application Server Liberty Version 8.5.5.1 in console mode, you might receive warning messages similar to the following example:

```
No "conClass" attribute in "commonPanel" element of panel com.ibm.was.liberty.userdata.panel.UserData in com.ibm.was.liberty.userdata.panel.
```

This problem is caused by an IBM Installation Manager API which is deprecated in Version 1.6.2. These warning messages can be ignored. No action is required.

## About this task

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

Perform one of these procedures to install, update, rollback, or uninstall the offering using Installation Manager.

**Note:** Before using Installation Manager to install a offering, you might want to back up your Installation Manager configuration using the instructions in the IBM Installation Manager Information Center if the possibility of corruption is a concern.

## Procedure

- “Installing Liberty on IBM i operating systems using response files” on page 794
- “Installing Liberty on IBM i operating systems using the command line” on page 790
- “Installing Liberty remotely on IBM i operating systems using the **iRemoteInstall** command” on page 800
- “Adding and removing features from Liberty on IBM i operating systems using response files” on page 803
- “Installing Liberty interim fixes on IBM i operating systems using the command line” on page 806
- “Uninstalling Liberty interim fixes from IBM i operating systems using the command line” on page 817
- “Installing Liberty fix packs on IBM i operating systems using response files” on page 814

- “Installing Liberty fix packs on IBM i operating systems using the command line” on page 809
- “Uninstalling Liberty fix packs from IBM i operating systems by using response files” on page 819
- “Uninstalling Liberty fix packs from IBM i operating systems using the command line” on page 818
- “Uninstalling Liberty from IBM i operating systems using response files” on page 806
- “Uninstalling Liberty from IBM i operating systems using the command line” on page 805

## Results

- The following locations are the defaults for Installation Manager files on IBM i systems:
  - **Installation location:** /QIBM/ProdData/InstallationManager
  - **Agent data location:** /QIBM/UserData/InstallationManager
  - **Registry:** /QIBM/InstallationManager/.ibm/registry/InstallationManager.dat
- Logs are located in the logs directory of Installation Manager's agent data location. For example:

/QIBM/UserData/InstallationManager/logs

The main log files are time-stamped XML files in the logs directory, and they can be viewed using any standard web browser.

## Installing Liberty on IBM i operating systems using the command line

You can install WebSphere Application Server Liberty Core on IBM i operating systems using the Installation Manager command line.

### Before you begin

Prepare for the installation before using this procedure. See Preparing the operating system for installation on IBM i for more information.



8.5.5.6

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

**Important:** Before installing WebSphere Application Server Liberty Core, you must read the license agreement that you can find with the product files. Signify your acceptance of the license agreement by specifying **-acceptLicense** in the command as described in this topic.

**Install Installation Manager** on the system onto which you want to install the product.

- If you want to use the Installation Manager that comes with this product, perform the following actions:
  1. Obtain the necessary files.
 

There are three basic options for obtaining and installing Installation Manager and the product.

    - **Access the physical media, and use local installation**

You can access the product repositories on the media.

      - a. Install Installation Manager on your system.
 

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.
      - b. Use Installation Manager to install the product from the product repositories on the media.
    - **Download the files from the Passport Advantage site, and use local installation**

Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

      - a. Download the files from the Passport Advantage site.

- b. Install Installation Manager on your system.

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.

- c. Use Installation Manager to install the product from the downloaded repositories.

– **Access the live repositories, and use web-based installation**

If you have a Passport Advantage ID and password, you can install the product from the web-based repositories.

- a. Install Installation Manager on your system.

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.

- b. Use Installation Manager to install the product from the web-based repository located at

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85>

**Note:** This location does not contain a web page that you can access using a web browser. This is a remote web-based repository location that you must specify for the value of the `-repositories` parameter so that the `imcl` command can access the files in this repository to install the product.

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

**Note:** If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

2. Choose three separate locations for Installation Manager's binaries, runtime data (agent data), and shared data locations.
3. Install Installation Manager using the Installation Manager command line.
  - a. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
  - b. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
  - c. Make sure that the `umask` is set to 022.

To verify the `umask` setting, issue the following command:

```
umask
```

To set the `umask` setting to 022, issue the following command:

```
umask 022
```

- d. Change to the location containing the Installation Manager installation files, and run the following command:

```
installc -acceptLicense -log log_file_path_and_name
```

**Notes:**

- For more information on installing Installation Manager, see the IBM Installation Manager Information Center.
- Use only the **installc** command to install Installation Manager.
- If you already have a version of Installation Manager installed on your system and you want to use it to install and maintain the product, obtain the necessary product files.

There are three basic options for installing the product.

– **Access the physical media, and use local installation**

You can access the product repositories on the media. Use Installation Manager to install the product from the product repositories on the media.

– **Download the files from the Passport Advantage site, and use local installation**

Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

1. Download the product repositories from the Passport Advantage site.
2. Use Installation Manager to install the product from the downloaded repositories.

– **Access the live repositories, and use web-based installation**

If you have a Passport Advantage ID and password, you can use Installation Manager to install the product from the web-based repositories. Use Installation Manager to install the product from the web-based repository located at

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85>

**Note:** This location does not contain a web page that you can access using a web browser. This is a remote web-based repository location that you must specify for the value of the **-repositories** parameter so that the **imcl** command can access the files in this repository to install the product.

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

**Note:** If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

## About this task

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

## Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the **imutilsc** command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Choose three separate locations for the product's binaries, runtime data (agent data), and shared data locations.
3. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
4. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
5. Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

6. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
7. Use the **imcl** command to install the product.




```
./imcl install com.ibm.websphere.liberty.v85_offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory installation_directory
-sharedResourcesDirectory shared_directory
-preferences preference_key=value
-properties property_key=value
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

### Tips:


- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `product_name/lafiles` subdirectory of the installation image or repository for this offering.
- You can install a list of features that are separated by commas.
  - Embeddable EJB container and JPA client (`embeddablecontainer`)  
This option installs the embeddable EJB container and JPA client.  
The embeddable EJB container is a Java Archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.  
The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

### Notes:

- If no features are specified, the default feature (`embeddablecontainer`) is installed.
-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.  
To install assets from the IBM WebSphere Liberty Repository, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later.  
If you want to install Liberty Repository features, specify the short names or symbolic names on the **user.feature** option of the **-properties** parameter. Multiple feature names are separated with double commas. The following example installs the Portlet Container and Portlet Serving features:

```
-properties user.feature=portlet-2.0,,portletserving-2.0,user.accept.license=true
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

-  **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` add-on, or install the individual features that you need from the Liberty Repository. See the following topics for more information:


- Installing Liberty Repository assets
- Liberty features

You can install the complete set of Extended Programming Model features by specifying the **user.addon** option:

```
-properties user.addon=extendedPackage-1.0,user.accept.license=true
```

If you upgrade WebSphere Application Server Liberty Version 8.5.5.3 or previous versions that contain the `extprogmodels` feature, Installation Manager automatically installs the `extendedPackage-1.0` add-on. You must specify the **user.accept.license** option:

```
-properties user.accept.license=true
```

-  **8.5.5.6** You can also install assets from instances of the Liberty Asset Repository Service or local directory-based repositories. For more information about these asset repositories, see “Installing assets using Installation Manager” on page 872. Add the repository on the **-repositories** parameter. The repositories are accessed in the order that they are specified. By default, the Liberty Repository is the last of the repositories that are accessed during installation. To

disable access to the Liberty Repository, on the **-properties** parameter, set the **user.useLibertyRepository** option to false. 8.5.5.8

```
./imcl install com.ibm.websphere.liberty.v85
-properties user.useLibertyRepository=false,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory /QIBM/ProdData/Liberty -acceptLicense
-repositories /QIBM/LibertyProductRepo,https://your_onprem_asset_repo_url,/QIBM/LocalAssetRepo,/QIBM/LocalAssetRepo2.zip
-sharedResourcesDirectory /QIBM/UserData/InstallationManager/IMShared
-showProgress
```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
./imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the **-installFixes** argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the **-installFixes** option defaults to all.
  - If the offering version is specified, the **-installFixes** option defaults to none.
- For initial installations, it is a good practice to specify the *user\_data\_root*; otherwise, the default value for the *user\_data\_root*, /QIBM/UserData/WebSphere/AppServer/V85/LibertyCore, is used. Use the *was.install.os400.profile.location* property to specify the *user\_data\_root*. If the *user\_data\_root* is to be /QIBM/UserData/WebSphere/AppServer/V85/Liberty, for example, specify **-properties was.install.os400.profile.location=/QIBM/UserData/WebSphere/AppServer/V85/Liberty** on the **imcl** installation command.
- The program might write important post-installation instructions to standard output.

For more information on using the **imcl** command to install the product, see the IBM Installation Manager Information Center.

## Example

The following example uses the **imcl** command to install Websphere Application Server LibertyCore:

```
./imcl install com.ibm.websphere.liberty.v85
-repositories https://downloads.mycorp.com:8080/WAS_85_repository
-installationDirectory /QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore
-properties was.install.os400.profile.location=/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore
-sharedResourcesDirectory /QIBM/UserData/InstallationManager/IMShared
-secureStorageFile $HOME/WASFiles/temp/credential.store -masterPasswordFile $HOME/WASFiles/IM/master_password_file.txt
-acceptLicense
```

## Installing Liberty on IBM i operating systems using response files

You can install WebSphere Application Server Liberty on IBM i operating systems using Installation Manager response files.

### Before you begin

Prepare for the installation before using this procedure. See Preparing the operating system for installation on IBM i for more information.



8.5.5.6

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

Before you install WebSphere Application Server, ensure that your user profile has \*ALLOBJ and \*SECADM special authorities.

**Install Installation Manager** on the system onto which you want to install the product.

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

- If you want to use the Installation Manager that comes with this product, perform the following actions:

1. Obtain the necessary files.

There are three basic options for obtaining and installing Installation Manager and the product.

- **Access the physical media, and use local installation**

You can access the product repositories on the media.

- a. Install Installation Manager on your system.

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.

- b. Use Installation Manager to install the product from the product repositories on the media.

- **Download the files from the Passport Advantage site, and use local installation**

Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

- a. Download the files from the Passport Advantage site.

- b. Install Installation Manager on your system.

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.

- c. Use Installation Manager to install the product from the downloaded repositories.

- **Access the live repositories, and use web-based installation**

If you have a Passport Advantage ID and password, you can install the product from the web-based repositories.

- a. Install Installation Manager on your system.

You can install Installation Manager using the media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager download website.

- b. Use Installation Manager to install the product from the web-based repository located at

<http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85>

**Note:** This location does not contain a web page that you can access using a web browser. This is a remote web-based repository location that you must specify in the response file so that the installation can access the files in this repository.

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

**Note:** If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

2. Install Installation Manager.

- a. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
- b. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
- c. Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

- d. Change to the temporary directory where you unpacked the Installation Manager files.
- e. Run the following command in the temporary folder:

```
installc -acceptLicense -log log_file_path_and_name
```

#### Notes:

- For more information on installing Installation Manager, see the IBM Installation Manager Information Center.
- Use only the **installc** command to install Installation Manager.
- If you already have a version of Installation Manager installed on your system and you want to use it to install and maintain the product, obtain the necessary product files.

There are three basic options for installing the product.

- **Access the physical media, and use local installation**

You can access the product repositories on the media. Use Installation Manager to install the product from the product repositories on the media.

- **Download the files from the Passport Advantage site, and use local installation**

Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

1. Download the product repositories from the Passport Advantage site.
2. Use Installation Manager to install the product from the downloaded repositories.

- **Access the live repositories, and use web-based installation**

If you have a Passport Advantage ID and password, you can use Installation Manager to install the product from the web-based repositories. Use Installation Manager to install the product from the web-based repository located at

```
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
```

**Note:** This location does not contain a web page that you can access using a web browser. This is a remote web-based repository location that you must specify in the response file so that the installation can access the files in this repository.

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

**Note:** If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

## Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the **imutilsc** command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store

credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
4. Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

5. Use a response file to install the product.

Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the product. For example:

```
./imcl -acceptLicense
input $HOME/WASFiles/temp/install_response_file.xml
-log $HOME/WASFiles/temp/install_log.xml
-secureStorageFile $HOME/WASFiles/temp/credential.store -masterPasswordFile $HOME/WASFiles/master_password_file.txt
```

#### Notes:

- The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `product_name/lafiles` subdirectory of the installation image or repository for this offering.
- `/QIBM/ProdData/InstallationManager` is the default installation location for Installation Manager files on IBM i systems.
- The program might write important post-installation instructions to standard output.

Read the IBM Installation Manager Information Center for more information.

## Example

The following is an example of a response file for installing the product with no optional features into the `/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore` directory using a web-based repository located at <http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85>.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
 <repository location='http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85' />
</server>
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
 <data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore' />
 <data key='user.import.profile' value='false' />
</profile>
<install modify='false'>
 <offering profile='WebSphere Liberty V8.5'
 features='' id='com.ibm.websphere.liberty.v85' />
</install>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache'
 value='/QIBM/UserData/InstallationManager/IMShared' />
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
</agent-input>
```

#### Tips:

- Make sure that the repository location points to the web-based or local product repository. For example:

```
<repository location='https://downloads.mycorp.com:8080/WAS_85_repository'/>
```

- The following line from the example specifies the default value of the profile location for IBM i:

```
<data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore'/>
```

To override this default location, specify a different location

- The following line from the example specifies the default value of the shared resources directory for IBM i:

```
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='/QIBM/UserData/InstallationManager/IMShared'/>
```

To override this default location, specify a different location

**Note:** There is only one shared resources directory for Installation Manager. If there has been an installation on the system in the past, it will use that shared resources directory and not the one specified in the response file.

- To disable remote searches for updates in the response file, set the following preferences to false:
  - `offering.service.repositories.areUsed`  
Used for searching remote repositories for updates to installed offerings
  - `com.ibm.cic.common.core.preferences.searchForUpdates`  
Used for searching for updates to Installation Manager

For example:

```
<preference value='false' name='offering.service.repositories.areUsed'/>
<preference value='false' name='com.ibm.cic.common.core.preferences.searchForUpdates'/>
```

You can find more details on silent preference keys in the IBM Installation Manager Information Center.

- To install more than one instance of an offering, you must make the profile ID of each additional instance unique. For example:

```
<offering profile='WebSphere Liberty V8.5 - Another User's WAS Liberty CORE'
 features='' id='com.ibm.websphere.liberty.v85'/>
```

This must be changed in both places that specify the profile ID in the response file.

Here are some examples of changes that you could make to manipulate this response file to perform alternative actions.

- To alter the location of the installation, simply change the installation location. For example:

Replace

```
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
```

with

```
<profile id='WebSphere Liberty V8.5' installLocation='/home/user/IBM/WebSphere/AppServer/V85/Server'>
```

- To install from a local repository instead of the live remote repository, replace the repository location. For example:

Replace

```
<repository location='http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85'/>
```

with

```
<repository location='/home/user/repositories/WAS85/local-repositories'/>
```

- To add the optional features, add each desired feature in the offering as an entry in a comma-separated list.

In the following list, the offering IDs to be used in the response files are enclosed in parentheses:

- Embeddable EJB container and JPA client (`embeddablecontainer`)

This option installs the embeddable EJB container and JPA client.

The embeddable EJB container is a Java Archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

#### Notes:

- If no features are specified, the default feature (`embeddablecontainer`) is installed.


For example, to install the Embeddable EJB container:

Replace

```
<offering profile='WebSphere Liberty V8.5'
 features='' id='com.ibm.websphere.liberty.v85'/>
```

with

```
<offering profile='WebSphere Liberty V8.5'
 features='embeddablecontainer' id='com.ibm.websphere.liberty.v85'/>
```

-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install Liberty Repository assets, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later. Previous versions of Installation Manager do not have the option to install Liberty Repository assets. If you use a response file and did not update Installation Manager to Version 1.6.2 or later, the assets that you specify in the response file are ignored during installation.

If you want to install additional features, specify two extra data key elements in your response file. You can use either the symbolic name or the short name.


The following example installs the Portlet Container and Portlet Serving features using the symbolic name.

```
<data key='user.feature' value='com.ibm.websphere.appserver.portlet-2.0,,com.ibm.websphere.appserver.portletserving-2.0'/>
<data key='user.accept.license' value='true'/>
```

The following example installs the Portlet Container and Portlet Serving features using the short name:

```
<data key='user.feature' value='portlet-2.0,,portletserving-2.0'/>
<data key='user.accept.license' value='true'/>
```


**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

-  **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features

The following example installs the Extended Programming Models using the `user.addon` parameter and the Portlet Container and Portlet Serving features using the `user.feature` parameter with short names:

```
<data key='user.addon' value='extendedPackage-1.0'/>
<data key='user.feature' value='portlet-2.0,,portletserving-2.0'/>
<data key='user.accept.license' value='true'/>
```

-  **8.5.5.6** You can also install assets from an instance of the Liberty Asset Repository Service or local directory-based repositories. For more information, see “Installing assets using Installation Manager” on page 872. Add the repository on `repository` elements. If Installation Manager does not recognize the repository, point directly to the `repository.config` file. When you install assets, the repositories are accessed in the order that you specify them.

```

<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85" />
<repository location="https://your_onprem_asset_repo_url" />
<repository location="/QIBM/LocalAssetRepo" />

```

8.5.5.8

<repository location="/QIBM/LocalAssetRepo2.zip" /> </server>By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, set the **user.useLibertyRepository** parameter to false:

```

<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portlet-serving-2.0' />
<data key='user.useLibertyRepository' value='false' />

```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

## Installing Liberty remotely on IBM i operating systems using the **iRemoteInstall** command

You can use the **iRemoteInstall** command to install IBM Installation Manager or WebSphere Application Server Liberty Core from a Windows workstation to a remote target IBM i system.

### Before you begin

Prepare for the installation before using this procedure. See [Preparing the operating system for installation on IBM i](#) for more information.

The product offering repository files or the IBM Installation Manager for IBM i installation kit compressed file must be available on the Windows system.

**Important:** You must set your `JAVA_HOME` environment variable to your IBM Installation Manager JRE home before running the command directly from the media.

### Restrictions:

- The **iRemoteInstall** command does not support credential-storage files used to pass confidential information. You must use the physical media or download the installation files to your local system.
- The **iRemoteInstall** command does not support the use of response files.

### About this task

**Note:** By running this script, you accept the terms of the product license. The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or `offering_name/lafiles` subdirectory of the installation image or repository for this offering.

#### Location of the **iRemoteInstall** command:

The **iRemoteInstall** command is located in the following directory when it has been installed as part of the WebSphere Customization Toolbox:

```
wct_root/Remote_Installation_Tool_for_IBM_i
```

**Tip:** A version of this utility that is current when the product is released is also available on the media or installation image. You can run the command directly from the media connected to a Windows system to install the offering on a remote target IBM i system. This version of the utility is located at the following location:

```
media_root\Remote_Installation_Tool_for_IBM_i\iRemoteInstall.bat
```

where `media_root` is the root directory of the media or installation image containing the product or supplements.

#### Syntax of the **iRemoteInstall** command:



```
iRemoteInstall.bat
-hostname i5_hostname
-username user_login_name
-password user_login_password
-iminstitkit im_install_kit_file_path_and_name | -wasoid was_offering_id
-wasrepoloc was_install_file_location
-appdataloc im_agent_data_location
-wasinstloc was_install_location
-wassharedloc was_shared_location
-features feature_ID_1,feature_ID_2, . . .
-properties key=value,key=value, . . .
-log log_file_path_and_name
-trace
-version
-help
```

### Parameters of the iRemoteInstall command:

**-hostname *i5\_hostname***

Specifies the host name of the target IBM i machine to which Installation Manager or WebSphere Application Server Liberty Core is going to be installed

This parameter is required.

**-username *user\_login\_name***

Specifies the login name of the user who is performing the Installation Manager or WebSphere Application Server Liberty Core remote installation

This user must be a valid user for the target IBM i system with \*ALLOBJ and \*SECADM special authorization.

**-password *user\_login\_password***

Specifies the login password of the user specified in **-username**

**-iminstitkit *im\_install\_kit\_file\_name***

Specifies the location of the Installation Manager for IBM i installation kit

You must include the path if it is not in the same directory as the command.

This parameter is required.

**-wasoid *was\_offering\_id***

Specifies the ID of the WebSphere Application Server Liberty offering being installed

Example values are base, nd, express, etc. This parameter is not case sensitive.

The value to use can be found in the product offering ID. If the offering ID is com.ibm.websphere.liberty.XXX.v85, for example, the **-wasoid** value should be liberty.XXX.

**-wasrepoloc *was\_install\_file\_location***

Specifies the location of the WebSphere Application Server Liberty Core installation repository

This option must be specified if the **-wasoid** parameter is specified.

**-appdataloc *im\_agent\_data\_location***

Specifies the location of the Installation Manager agent data

If no value is specified for this parameter, it is set to the default value of /QIBM/UserData/InstallationManager.

**-wasinstloc *was\_install\_location***

Specifies the location of the WebSphere Application Server Liberty Core installation

If no value is specified for this parameter, it is set to the default value of /QIBM/WAS85/Liberty.

**-wassharedloc *was\_shared\_location***

Specifies the location of the WebSphere Application Server Liberty Core shared location

If no value is specified for this parameter, it is set to the default value of /QIBM/WAS85/Liberty\_Shared.

**-features *feature\_ID\_1,feature\_ID\_2, . . .***

Specifies the features to be installed

The feature IDs must be separated by commas (,).

**Tip:** If no features are specified, the default feature (`embeddablecontainer`) is installed.

**-properties *key=value,key=value, . . .***

Specifies package-group (profile) properties

**-log *log\_file\_path\_and\_name***

Turns on the log, and sends all messages to the specified file and location

The path can be absolute (`c:\temp\mylog.log` for example) or relative (`..\mylog.log` for example).

Because you can append multiple installation actions into the same log, the actual name of a log file that is generated is `log_file_path_and_name.x.log`, where *x* is the number of the log file from 0 to 29. The maximum log file size is approximately 10 MB; and the maximum number of log files generated is 30.

**-trace**

Provides trace output of what the command checks and what the command discovers

**-version**

Displays the version information for the command

**-help**

Displays usage information for the command

## Procedure

1. Log in to the IBM i machine using the IBM Personal Communications tool, or telnet with TN5250 to the IBM i machine.
2. If TCP/IP is not started or if you do not know if TCP/IP is started, enter the following command on the Control Language (CL) command line:

```
STRTCP
```

3. Verify that the host server jobs are started on your IBM i server.

The host server jobs allow the installation code to run on IBM i.

Enter the following command on the CL command line:

```
STRHOSTSVR SERVER(*ALL)
```

4. Verify that your user profile has `*ALLOBJ` and `*SECADM` special authorities.
5. Run the **iRemoteInstall** command in the temporary directory to install Installation Manager or Websphere Application Server Liberty Core.  
In order to install Websphere Application Server Liberty Core, Installation Manager must already be installed on the target system.
6. Verify the installation.
  - Check for error messages in the output from the **iRemoteInstall** command.
  - Look for errors in the installation log.

## Example

Here is an example of installing IBM Installation Manager with the **iRemoteInstall** command:

```
./iRemoteInstall
-hostname iserver1.somedomain.com
-username wasadmin -password mypwd
-iminstkit E:\agent.installer.os400.motif.ppc_1.6.2000.20101206_0100.zip
```

Here is an example of installing WebSphere Application Server Liberty Core with the `iRemoteInstall` command:

```
./iRemoteInstall
-hostname iserver1.somedomain.com
-username wasadmin -password mypwd
-wasoid liberty.CORE
-wasrepoloc E:\repository
```

## Verifying the installation

You can verify successful installation of the offering using the capabilities of IBM Installation Manager.

### Procedure

- To verify installation of the offering, you can use Installation Manager to find the offering in the list of installed packages.

Change the directory to the `eclipse/tools` subdirectory of the Installation Manager binaries location, and run this command:

```
./imcl listInstalledPackages
```

This will display a list indicating which packages this Installation Manager has installed. For example:

```
com.ibm.websphere.liberty.v85_8.5.5.20110203_0234
```

- If an installation was successful, the `installed.xml` file should contain a location element for the installed offering.

For example, the following file:

```
installation_manager_root/properties/version/installed.xml
```

should contain something like this:

```
<location id="IBM WebSphere Application Server Liberty Core V8.5" kind="product" path="/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore"> </location>
```

- If you used the Installation Manager `-log` option during installation, you can verify that the resulting log file does not contain any errors.

If you used the following command to install the offering silently for example:

```
./imcl -acceptLicense
input $HOME/WASFiles/liberty/temp/install_response_file.xml
-log $HOME/WASFiles/liberty/temp/install_log.xml
-secureStorageFile $HOME/WASFiles/liberty/temp/credential.store -masterPasswordFile $HOME/WASFiles/liberty/master_password_file.txt
```

and the installation was successful, the `install_log.xml` file should contain something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

## Adding and removing features from Liberty on IBM i operating systems using response files

You can install and remove WebSphere Application Server Liberty Core features using Installation Manager response files.

### About this task

Perform this procedure to use Installation Manager to install or remove a feature silently using a response file.

Like other Installation Manager operations, you can invoke a modification using the `imcl` command-line tool. Go to the IBM Installation Manager Information Center for more information.

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

**Optional features:** In the following list of optional features, the names to be used in the response files are enclosed in parentheses:

- Embeddable EJB container and JPA client (`embeddablecontainer`)

This option installs the embeddable EJB container and JPA client.

The embeddable EJB container is a Java Archive (JAR) file that you can use to run enterprise beans in a standalone Java Platform, Standard Edition (SE) environment. You can run enterprise beans using this embeddable container outside the application server. The embeddable EJB container is a part of the EJB 3.1 specification and is primarily used for unit testing enterprise beans business logic.

The JPA client can be used with the embeddable EJB container to provide Java Persistence API capability in a Java SE environment.

## Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. On a CL command line, run the `STRQSH` command to start the Qshell command shell.
4. Use a response file to install or remove a feature.

Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and modify the product. For example:

```
./imcl
input $HOME/WASFiles/temp/modify_response_file.xml
-log $HOME/WASFiles/temp/modify_log.xml
-secureStorageFile $HOME/WASFiles/temp/credential.store -masterPasswordFile $HOME/WASFiles/master_password_file.txt
```

**Note:** The program might write important post-installation instructions to standard output.

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

## Example

- Here are examples of response files for modifying the features in an installation:
  - Here is a response file that adds the Embeddable EJB container and JPA client to an existing product that is installed in the `/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore` directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
 <repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85' />
</server>
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
 <data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore' />
</profile>
<install modify='true'>
 <offering profile='WebSphere Liberty V8.5' features='embeddablecontainer' id='com.ibm.websphere.liberty.v85' />
</install>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='/QIBM/UserData/InstallationManager/IMShared' />
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
```

```

<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
</agent-input>

```

- To alter this response file to remove a feature, simply change the `install` tags to `uninstall`. Here is the same response file modified to remove the Embeddable EJB container and JPA client:

```

<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
 <repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85'/>
</server>
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
 <data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore'>
</profile>
<uninstall modify='true'>
 <offering profile='WebSphere Liberty V8.5' features='embeddablecontainer' id='com.ibm.websphere.liberty.v85'/>
</uninstall>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='/QIBM/UserData/InstallationManager/IMShared'/>
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
</agent-input>

```

- To combine adding and removing features using a single response file, add both an `install` action and an `uninstall` action.

- Here is an example of the `imcl` command for modifying the features in an installation:

```

./imcl modify com.ibm.websphere.liberty.v85
-addFeatures embeddablecontainer
-repositories http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85
-installationDirectory /QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore
-secureStorageFile /var/credential.store -masterPasswordFile /var/master_password_file.txt

```

## Uninstalling Liberty from IBM i operating systems using the command line

You can use Installation Manager to uninstall WebSphere Application Server Liberty Core using the Installation Manager command line (`imcl`).

### Procedure

1. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
4. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
5. Use the `imcl` command to uninstall the product.

For example:

```

./imcl uninstall com.ibm.websphere.liberty.v85,optional_feature_ID
-installationDirectory installation_directory

```

You can remove a list of features that are separated by commas. If a list of features is not specified, the entire product is uninstalled.

For more information on using the `imcl` command to uninstall the product, see the IBM Installation Manager Information Center.

6. Optional: Uninstall IBM Installation Manager.

**Important:** Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

For more information on uninstalling Installation Manager, see the IBM Installation Manager Information Center.

## Example

Here is an example of using the **imcl** command to uninstall WebSphere Application Server:

```
./imcl uninstall com.ibm.websphere.liberty.v85
-installationDirectory /QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore
```

## Uninstalling Liberty from IBM i operating systems using response files

You can uninstall WebSphere Application Server Liberty Core using Installation Manager response files.

### About this task

Using Installation Manager, you can work with response files to uninstall the product.

### Procedure

1. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
4. Use a response file to uninstall the product.

From a command line on each of the systems from which you want to uninstall the product, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager and use a response file that you created to uninstall the product. For example:

```
./imcl
input $HOME/WASFiles/temp/uninstall_response_file.xml
-log $HOME/WASFiles/temp/uninstall_log.xml
```

Here is an example of what the response file might contain:

```
<agent-input>
<uninstall>
<offering profile="WebSphere Liberty V8.5"/>
</uninstall>
</agent-input>
```

Go to the IBM Installation Manager Information Center for more information.

5. Optional: Uninstall IBM Installation Manager.

**Important:** Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

Read the IBM Installation Manager Information Center for information about using the uninstall script to perform this procedure.

## Installing Liberty interim fixes on IBM i operating systems using the command line

Fix packs contain bundled service to bring WebSphere Application Server Liberty for IBM i up to a new level. Interim fixes provide corrective service for specific known problems. You can use the IBM Installation Manager command-line function to update the offering with the fixes that are available for your service level of WebSphere Application Server Liberty for IBM i.

### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server Liberty for IBM i. The most current information is available from the IBM Software Support Center and Fix Central.

IBM Installation Manager is used to apply maintenance to WebSphere Application Server Liberty for IBM i.

### About this task

Use this procedure whenever you want to apply a new interim fix to your system.

**Tip:** You can also install interim fixes using silent response files with Installation Manager. For information on creating and using response files, read the IBM Installation Manager Information Center.

**Restriction:** You cannot use the `iRemoteInstall` command to install an interim fix.

## Procedure

1. For a list of interim fixes that are available for WebSphere Application Server Liberty for IBM i and specific information about each interim fix, perform the following actions.
  - a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server Liberty for IBM i** as the product.
  - d. Select the version of the offering to be updated.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. **Recommendation:** Make a list of the names of the interim fixes that you would like to install.
2. Update WebSphere Application Server Liberty for IBM i with the interim fixes using one of the following procedures.

- **Access the live service repository that contains the fixes, and use web-based updating.**

Use Installation Manager on your local system to update WebSphere Application Server Liberty for IBM i with the interim fixes from the live web-based service repositories.

- For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are listed in Online product repositories for WebSphere Application Server offerings.
- These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the offering.

To install an interim fix from a service repository, perform the following actions:

- a. If you do not already have Installation Manager credential-storage and master-password files containing your IBM software user ID and password, create files that will allow you to access the repository.

**Note:** These are the credentials that you use to access protected IBM software websites.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

- b. Sign on to the IBM i system with a user profile that has `*ALLOBJ` and `*SECADM` special authorities.
- c. Stop all servers and applications on the WebSphere Application Server Liberty for IBM i installation that is being updated.

- d. On a CL command line, run the STRQSH command to start the Qshell command shell.
- e. Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

- f. Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.  
On IBM i systems, the root directory for the Installation Manager is */QIBM/ProdData/InstallationManager*.

- g. Install the interim fix.

```
./imcl install interim_fix_name
-installationDirectory offering_installation_location
-repositories repository_URL
-secureStorageFile storage_file -masterPasswordFile master_password_file
```

- h. **Optional:** List all installed packages to verify the installation:

```
./imcl listInstalledPackages -long
```

- **Download the files that contain the fixes from Fix Central, and use local updating.**

You can download compressed files that contain the fixes from Fix Central. Each compressed fix file contains an Installation Manager repository for the fix and usually has a .zip extension. After downloading the fix files, you can use Installation Manager to update WebSphere Application Server Liberty for IBM i with the interim fixes.

- a. To download the interim fixes, perform the following actions:

- 1) Go to Fix Central.
- 2) Select **WebSphere** as the product group.
- 3) Select **WebSphere Application Server Liberty for IBM i** as the product.
- 4) Select the version of the offering to be updated.
- 5) Select your operating system as the platform, and click **Continue**.
- 6) Select **Browse for fixes**, and click **Continue**.
- 7) Select the interim fixes that you want to download, and click **Continue**.
- 8) Select your download options, and click **Continue**.
- 9) Click **I agree** to agree to the terms and conditions.
- 10) Click **Download now** to download the interim fixes.
- 11) Transfer the compressed fix files in binary format to the IBM i system on which they will be installed.

- b. To install an interim fix from a downloaded file, perform the following actions:

- 1) Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
- 2) Stop all servers and applications on the WebSphere Application Server Liberty for IBM i installation that is being updated.
- 3) On a CL command line, run the STRQSH command to start the Qshell command shell.
- 4) Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```



- 5) Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.

On IBM i systems, the root directory for the Installation Manager is */QIBM/ProdData/InstallationManager*.

- 6) Install the interim fix.

```
./imcl install interim_fix_name
-installationDirectory offering_installation_location
-repositories compressed_file
```

- 7) **Optional:** List all installed packages to verify the installation:

```
./imcl listInstalledPackages -long
```

## Installing Liberty fix packs on IBM i operating systems using the command line

Product fix packs contain bundled service to bring WebSphere Application Server Liberty Core up to a new product level. Interim fixes provide corrective service for specific known problems. You can use the IBM Installation Manager command-line function to update the product with the fixes that are available for your service level of WebSphere Application Server Liberty Core installation.

### Before you begin

Contact the IBM Software Support Center for information about updates for WebSphere Application Server for IBM i. The most current information is available from the IBM Software Support Center and Fix Central.



8.5.5.6

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

**Tip:** As an alternative to the procedure that is described in this article, Installation Manager allows you to use the **updateAll** command in a response file or on the command line to search for and update all installed packages. Use this command only if you have full control over which fixes are contained in the targeted repositories. If you create and point to a set of custom repositories that include only the specific fixes that you want to install, you should be able to use this command confidently. If you enable searching service repositories or install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install using the **-installFixes** option with the **install** command on the command line or the **installFixes** attribute in a response file.

### About this task

- | **New:** Fix pack 16.0.0.2 is the next fix pack after 8.5.5.9. You can continue to use the same Version 8.5 Installation Manager repositories and offering IDs to install or update to 16.0.0.2, or you can use the new versionless repositories and offerings. For more information about fix pack 16.0.0.2, see What is new in Liberty in the new location of the latest Liberty documentation.
- You can also install fix packs using response files with Installation Manager. For information on creating and using response files, read “Installing Liberty fix packs on IBM i operating systems using response files” on page 814 and the IBM Installation Manager Information Center.
- You cannot use the `iRemoteInstall` command to install a fix pack.
- **8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

### Procedure

1. For a list of fixes that are available for WebSphere Application Server Liberty Core installation and specific information about each fix, perform the following actions.

- a. Go to Fix Central.
  - b. Select **WebSphere** as the product group.
  - c. Select **WebSphere Application Server** as the product.
  - d. Select the installed version.
  - e. Select your operating system as the platform, and click **Continue**.
  - f. Select **Browse for fixes**, and click **Continue**.
  - g. Click **More Information** under each fix to view information about the fix.
  - h. **Recommendation:** Make a list of the names of the fixes that you would like to install.
2. Update WebSphere Application Server LibertyCore installation with the fix pack using one of the following procedures.
- Access the live service repository that contains the fix pack, and use web-based updating. Use Installation Manager on your local system to update WebSphere Application Server Liberty Core with the interim fixes from the live web-based service repositories.
    - For the live service repositories, use the same URLs as those used for the generally available product-offering repositories during installation. These URLs are based on the following pattern:  
`http://www.ibm.com/software/repositorymanager/offering_ID`

where *offering\_ID* is the offering ID that you can find in WebSphere Application Server product offerings for supported operating systems.

  - These locations do not contain web pages that you can access using a web browser. They are remote web-based repository locations that you specify for Installation Manager so that it can maintain the product.
- To install a fix from a service repository, perform the following actions:
- a. If you do not already have Installation Manager credential-storage and master-password files containing your IBM software user ID and password, create files that will allow you to access the repository.
- Note:** These are the credentials that you use to access protected IBM software websites. For information on creating credential-storage and master-password files for Installation Manager, read the IBM Installation Manager Information Center.
- Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.
- b. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
- c. Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.
- d. On a CL command line, run the STRQSH command to start the Qshell command shell.
- e. Make sure that the umask is set to 022.  
To verify the umask setting, issue the following command:  

```
umask
```

To set the umask setting to 022, issue the following command:  

```
umask 022
```

- f. Change to the `Installation_Manager_binaries/eclipse/tools` directory, where `Installation_Manager_binaries` is the installation root directory for the Installation Manager. On IBM i systems, the root directory for the Installation Manager is `/QIBM/ProdData/InstallationManager`.
- g. Install the fix pack.

```
./imcl install offering_ID offering_version,optional_feature_ID
-repositories source_repository
-installationDirectory offering_installation_location
-secureStorageFile storage_file -masterPasswordFile master_password_file
-acceptLicense
```

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.


### Tips:

- The *offering\_ID* is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.
- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
./imcl listAvailablePackages -repositories source_repository
```

- You can also specify *none*, *recommended* or *all* with the **-installFixes** argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the **-installFixes** option defaults to *all*.
  - If the offering version is specified, the **-installFixes** option defaults to *none*.
- You can add a list of features that are separated by commas.


-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install assets from the IBM WebSphere Liberty Repository, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later.

If you want to install Liberty Repository features, specify the short names or symbolic names on the **user.feature** option of the **-properties** parameter. Multiple feature names are separated with double commas. The following example installs the Portlet Container and Portlet Serving features:

```
-properties user.feature=portlet-2.0,,portletserving-2.0,user.accept.license=true
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

 **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features

You can install the complete set of Extended Programming Model features by specifying the **user.addon** option:

```
-properties user.addon=extendedPackage-1.0,user.accept.license=true
```

If you upgrade WebSphere Application Server Liberty Version 8.5.5.3 or previous versions that contain the `extprogmodel's` feature, Installation Manager automatically installs the `extendedPackage-1.0` add-on. You must specify the `user.accept.license` option:

```
-properties user.accept.license=true
```



8.5.5.6

You can also install assets from instances of the Liberty Asset Repository Service or local directory-based repositories. For more information about these asset repositories, see “Installing assets using Installation Manager” on page 872. Add the repository on the `-repositories` parameter. The repositories are accessed in the order that they are specified. By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, on the `-properties` parameter, set the `user.useLibertyRepository` option to `false`.

8.5.5.8

```
./imcl install com.ibm.websphere.liberty.v85
-properties user.useLibertyRepository=false,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory /QIBM/ProdData/Liberty -acceptLicense
-repositories /QIBM/LibertyProductRepo,https://your_onprem_asset_repo_url,/QIBM/LocalAssetRepo,/QIBM/LocalAssetRepo2.zip
-sharedResourcesDirectory /QIBM/UserData/InstallationManager/IMShared
-showProgress
```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- If you obtained the fix pack by installing the WebSphere Application Server group PTF, you can use the local fix-pack repositories to install the fix pack.

For information about the local fix-pack repositories, see file `/QIBM/WAS/WASFixpacks/ReadmeV85.html` or `/QIBM/WAS/WASFixpacks/ReadmeV85.txt`.

- h. **Optional:** List all installed packages to verify the installation:

```
./imcl listInstalledPackages -long
```

- Download a file that contains the fix pack from Fix Central, and use local updating. You can download a compressed file that contains the fix pack from Fix Central. Each compressed fix file contains an Installation Manager repository for the fix pack and usually has a `.zip` extension. After downloading the fix file, you can use Installation Manager to update WebSphere Application Server Liberty Core with the fix pack.
  - a. To download the fix pack, perform the following actions:
    - 1) Go to Fix Central.
    - 2) Select **WebSphere** as the product group.
    - 3) Select **WebSphere Application Server** as the product.
    - 4) Select the installed version.
    - 5) Select your operating system as the platform, and click **Continue**.
    - 6) Select **Browse for fixes**, and click **Continue**.
    - 7) Select the fix pack that you want to download, and click **Continue**.
    - 8) Select your download options, and click **Continue**.
    - 9) Click **I agree** to agree to the terms and conditions.
    - 10) Click **Download now** to download the fix pack.
    - 11) Transfer the compressed fix file in binary format to the IBM i systems on which it will be installed.
    - 12) Extract the compressed repository file to a directory on your system.
  - b. To install a fix pack from a downloaded file, perform the following actions:
    - 1) Sign on to the IBM i system with a user profile that has `*ALLOBJ` and `*SECADM` special authorities.
    - 2) Stop all servers and applications on the WebSphere Application Server Liberty Core installation that is being updated.

- 3) On a CL command line, run the STRQSH command to start the Qshell command shell.
- 4) Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

- 5) Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager.

On IBM i systems, the root directory for the Installation Manager is */QIBM/ProdData/InstallationManager*.

- 6) Install the fix pack.

```
./imcl install offering_ID_offering_version,optional_feature_ID
-repositories location_of_expanded_files
-installationDirectory offering_installation_location
-acceptLicense
```

#### Tips:

- The *offering\_ID* is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.
- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
./imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the *-installFixes* argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the *-installFixes* option defaults to all.
  - If the offering version is specified, the *-installFixes* option defaults to none.
- You can add a list of features that are separated by commas.
- You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install assets from the IBM WebSphere Liberty Repository, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later.

If you want to install Liberty Repository features, specify the short names or symbolic names on the **user.feature** option of the **-properties** parameter. Multiple feature names are separated with double commas. The following example installs the Portlet Container and Portlet Serving features:

```
-properties user.feature=portlet-2.0,,portletserving-2.0,user.accept.license=true
```

8.5.5.5

For Version 8.5.5.5 and later, `user.accept.license=true` is not required.



8.5.5.4

Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features

You can install the complete set of Extended Programming Model features by specifying the **user.addon** option:

```
-properties user.addon=extendedPackage-1.0,user.accept.license=true
```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

If you upgrade WebSphere Application Server Liberty Version 8.5.5.3 or previous versions that contain the `extprogmodel`s feature, Installation Manager automatically installs the `extendedPackage-1.0` add-on. You must specify the **user.accept.license** option:

```
-properties user.accept.license=true
```



**8.5.5.6**

You can also install assets from instances of the Liberty Asset Repository Service or local directory-based repositories. For more information about these asset repositories, see “Installing assets using Installation Manager” on page 872. Add the repository on the **-repositories** parameter. The repositories are accessed in the order that they are specified. By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, on the **-properties** parameter, set the **user.useLibertyRepository** option to `false`.

**8.5.5.8**

```
imcl install com.ibm.websphere.liberty.v85
-properties user.useLibertyRepository=false,user.addon=extendedPackage-1.0,user.feature=portlet-2.0
-installationDirectory D:\IBM\Liberty -acceptLicense
-repositories D:\IBM\LibertyProductRepo,https://your_onprem_asset_repo_url,D:\IBM\LocalAssetRepo,D:\IBM\LocalAssetRepo2.zip
-sharedResourcesDirectory D:\IBM\IMShared
-showProgress
```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

- If you obtained the fix pack by installing the WebSphere Application Server group PTF, you can use the local fix-pack repositories to install the fix pack.

For information about the local fix-pack repositories, see file `/QIBM/WAS/WASFixpacks/ReadmeV85.html` or `/QIBM/WAS/WASFixpacks/ReadmeV85.txt`.

- 7) **Optional:** List all installed packages to verify the installation:

```
./imcl listInstalledPackages -long
```

## Installing Liberty fix packs on IBM i operating systems using response files

You can update WebSphere Application Server Liberty Core to a later version using Installation Manager response files.

### Before you begin



**8.5.5.6**

If you want to install Liberty assets from local directory-based repositories or an instance of the Liberty Asset Repository Service, configure the repositories. For more information about the Liberty asset repositories, see “Installing assets using Installation Manager” on page 872.

**Tip:** As an alternative to the procedure that is described in this article, Installation Manager allows you to use the **updateAll** command in a response file or on the command line to search for and update all installed packages. Use this command only if you have full control over which fixes are contained in the targeted repositories. If you create and point to a set of custom repositories that include only the specific fixes that you want to install, you should be able to use this command confidently. If you enable searching service repositories or install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install using the **-installFixes** option with the **install** command on the command line or the **installFixes** attribute in a response file.

## About this task

- | **New:** Fix pack 16.0.0.2 is the next fix pack after 8.5.5.9. You can continue to use the same Version 8.5
- | Installation Manager repositories and offering IDs to install or update to 16.0.0.2, or you can use the new
- | versionless repositories and offerings. For more information about fix pack 16.0.0.2, see What is new in
- | Liberty in the new location of the latest Liberty documentation.

**8.5.5.4** To install Version 8.5.5.4 and later of Liberty, you must have IBM Installation Manager Version 1.6.2 or later.

## Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the `imutilsc` command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the `-keyring` and `-password` options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the `-secureStorageFile` and `-masterPasswordFile` options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the `-keyring` and `-password` options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. On a CL command line, run the `STRQSH` command to start the Qshell command shell.
4. Use a response file to update the product.

Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and update the product. For example:

```
./imcl -acceptLicense
input $HOME/WASFiles/temp/update_response_file.xml
-log $HOME/WASFiles/temp/update_log.xml
-secureStorageFile $HOME/WASFiles/temp/credential.store -masterPasswordFile $HOME/WASFiles/master_password_file.txt
```

**Note:** The program might write important post-installation instructions to standard output.

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

## Example

The following is an example of a response file for updating WebSphere Application Server Liberty Core to a later version.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
 <repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85' />
</server>
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
 <data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/AppServer/V85/LibertyCore' />
</profile>
<install modify='false'>
 <offering profile='WebSphere Liberty V8.5' id='com.ibm.websphere.liberty.v85'
 version='8.5.5.20101025_2108' />
</install>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='/QIBM/UserData/InstallationManager/IMShared' />
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
```

```

<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
</agent-input>

```

### Tips:


- The profile ID (<profile . . . id='profile\_ID' . . . > and <offering . . . profile='profile\_ID' . . . >) can be found when you run the `imcl listInstallationDirectories -verbose` command from the `eclipse/tools` subdirectory in the directory where you installed Installation Manager. It is the same as the package group's name.
- The *version* is a specific version of the offering to install (8.5.5.20101025\_2108 for example). This specification is optional.
  - If *version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

```
./imcl listAvailablePackages -repositories source_repository
```

- You can also specify none, recommended or all with the `-installFixes` argument to indicate which interim fixes you want installed with the offering.
  - If the offering version is **not** specified, the `-installFixes` option defaults to all.
  - If the offering version is specified, the `-installFixes` option defaults to none.
- If you obtained the fix pack by installing the WebSphere Application Server group PTF, you can use the local fix-pack repositories to install the fix pack.

For information about the local fix-pack repositories, see file `/QIBM/WAS/WASFixpacks/ReadmeV8.html` or `/QIBM/WAS/WASFixpacks/ReadmeV8.txt`.

-  **8.5.5.2** You can specify additional assets to install from the Liberty Repository. For a list of Liberty Repository assets, see the downloads page on WASdev.net.

To install Liberty Repository assets, you must have access to the internet, and you must have IBM Installation Manager Version 1.6.2 or later. Previous versions of Installation Manager do not have the option to install Liberty Repository assets. If you use a response file and did not update Installation Manager to Version 1.6.2 or later, the assets that you specify in the response file are ignored during installation.

If you want to install additional features, specify two extra data key elements in your response file. You can use either the symbolic name or the short name.

The following example installs the Portlet Container and Portlet Serving features using the symbolic name.

```

<data key='user.feature' value='com.ibm.websphere.appserver.portlet-2.0,,com.ibm.websphere.appserver.portletserving-2.0' />
<data key='user.accept.license' value='true' />

```


The following example installs the Portlet Container and Portlet Serving features using the short name:

```

<data key='user.feature' value='portlet-2.0,,portletserving-2.0' />
<data key='user.accept.license' value='true' />

```

**8.5.5.5** For Version 8.5.5.5 and later, `user.accept.license=true` is not required.

 **8.5.5.4** Beginning with Version 8.5.5.4, the `extprogmodels` feature is no longer available. Instead, install the `extendedPackage-1.0` addon, or install the individual features that you need from the Liberty Repository. See the following topics for more information:

- Installing Liberty Repository assets
- Liberty features



The following example installs the Extended Programming Models using the **user.addon** parameter and the Portlet Container and Portlet Serving features using the **user.feature** parameter with short names:

```
<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portlet-serving-2.0' />
<data key='user.accept.license' value='true' />
```



8.5.5.6

You can also install assets from an instance of the Liberty Asset Repository Service or local directory-based repositories. For more information, see “Installing assets using Installation Manager” on page 872. Add the repository on repository elements. If Installation Manager does not recognize the repository, point directly to the repository.config file. When you install assets, the repositories are accessed in the order that you specify them.

```
<server>
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85" />
<repository location="https://your_onprem_asset_repo_url" />
<repository location="/QIBM/LocalAssetRepo" />
```

8.5.5.8

`<repository location="/QIBM/LocalAssetRepo2.zip" />` `</server>` By default, the Liberty Repository is the last of the repositories that are accessed during installation. To disable access to the Liberty Repository, set the **user.useLibertyRepository** parameter to false:

```
<data key='user.addon' value='extendedPackage-1.0' />
<data key='user.feature' value='portlet-2.0,,portlet-serving-2.0' />
<data key='user.useLibertyRepository' value='false' />
```

To learn more about the Liberty Repository and the assets it contains, see “Liberty Repository” on page 573.

•

**Note:** 8.5.5.3 If you are updating to Version 8.5.5.3 and previously installed Liberty Repository features but do not currently have a connection to the IBM WebSphere Liberty Repository, you cannot update using a response file. Instead, update the product by running the **imcl** command and specifying the **user.feature=""** parameter.

## Uninstalling Liberty interim fixes from IBM i operating systems using the command line

You can use the IBM Installation Manager command-line function to remove interim fixes from WebSphere Application Server Liberty Core.

### About this task

Use this procedure whenever you want to remove an interim fix from your system using the command line.

**Tip:** You can also uninstall interim fixes using silent response files with Installation Manager. For information on creating and using response files, read “Installing Liberty fix packs on IBM i operating systems using response files” on page 814 and the IBM Installation Manager Information Center.

### Procedure

1. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
2. Stop all servers and applications on the WebSphere Application Server Liberty Core installation.
3. On a CL command line, run the STRQSH command to start the Qshell command shell.
4. Make sure that the umask is set to 022.

To verify the umask setting, issue the following command:

```
umask
```

To set the umask setting to 022, issue the following command:

```
umask 022
```

5. Change to the *Installation\_Manager\_binaries/eclipse/tools* directory, where *Installation\_Manager\_binaries* is the installation root directory for the Installation Manager. On IBM i systems, the root directory for the Installation Manager is */QIBM/ProdData/InstallationManager*.
6. Uninstall the interim fix:
 

```
./imcl uninstall interim_fix_name
 -installationDirectory offering_installation_location
```
7. Optional: List all installed packages to verify the uninstallation.
 

```
./imcl listInstalledPackages -long
```

## Uninstalling Liberty fix packs from IBM i operating systems using the command line

You can roll back WebSphere Application Server Liberty Core to an earlier version using the Installation Manager command line.

### Before you begin

**Restriction:** In order to use this procedure, you must have Installation Manager Version 1.6 or later installed on your system.

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

### Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append */repository.config* at the end of the repository URL location if the **imutilsc** command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Sign on to the IBM i system with a user profile that has **\*ALLOBJ** and **\*SECADM** special authorities.
3. Stop all servers and applications on the installation that is being rolled back.
4. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
5. Change to the *eclipse/tools* subdirectory in the directory where you installed Installation Manager.
6. Use the **imcl** command to roll back the product.

```
./imcl rollback offering_ID_offering_version
 -repositories source_repository
 -installationDirectory installation_directory
 -preferences preference_key=value
 -properties property_key=value
 -secureStorageFile storage_file -masterPasswordFile master_password_file
 -acceptLicense
```

### Tips:

- The *offering\_ID* is the offering ID that is listed in WebSphere Application Server product offerings for supported operating systems.
- The *offering\_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to which to roll back (8.5.5.20110503\_0200 for example).
  - If *offering\_version* is **not** specified, the installation rolls back to the previously installed version of the offering and **all** interim fixes for that version are installed.
  - If *offering\_version* is specified, the installation rolls back to the specified earlier version of the offering and **no** interim fixes for that version are installed.
- If you previously installed Liberty Repository features and addons but do not have access to a Liberty repository when you rollback your installation, specify the following properties in the response file:  

```
-properties user.addon="",user.feature=""
```

Specifying these properties enables the product to rollback and uninstalls all features and addons.

For more information on using the **imcl** command, read the IBM Installation Manager Information Center.

7. Optional: List all installed packages to verify the roll back.

```
./imcl listInstalledPackages -long
```

## Uninstalling Liberty fix packs from IBM i operating systems by using response files

You can rollback WebSphere Application Server Liberty Core to an earlier version by using Installation Manager response files.

### Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

### Procedure

1. Optional: If the repository requires a username and password, create a credential-storage file to access this repository.

**Tip:** When creating a credential-storage file, append `/repository.config` at the end of the repository URL location if the **imutilsc** command is unable to find the URL that is specified.

**Note:** When you use Installation Manager Version 1.6.2 and later, you should use the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file. In versions of Installation Manager earlier than Version 1.6.2, the **-keyring** and **-password** options were used to access credentials in a keyring file. These options were deprecated in Version 1.6.2. There is no migration path from keyring files to storage files because of the differences in the file structures. For more information on using the **-secureStorageFile** and **-masterPasswordFile** options to store credentials in a credential-storage file, see the Installation Manager Version 1.6 documentation. For more information on using the **-keyring** and **-password** options to store credentials in a keyring file, see the Installation Manager Version 1.5 documentation.

2. Sign on to the IBM i system with a user profile that has \*ALLOBJ and \*SECADM special authorities.
3. Stop all servers and applications on the installation that is being rolled back.
4. On a CL command line, run the **STRQSH** command to start the Qshell command shell.
5. Use a response file to rollback the product.

Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and rollback the product. For example:

```
./imcl
input $HOME/WASFiles/temp/rollback_response_file.xml
-log $HOME/WASFiles/temp/rollback_log.xml
-secureStorageFile $HOME/WASFiles/temp/credential.store -masterPasswordFile $HOME/WASFiles/master_password_file.txt
```

**Note:** The program might write important post-installation instructions to standard output.

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

#### 6. Optional: List all installed packages to verify the rollback.

```
./imcl listInstalledPackages -long
```

## Example

The following is an example of a response file for rolling back the product to an earlier version.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
 <repository location='https://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.v85' />
</server>
<profile id='WebSphere Liberty V8.5' installLocation='/QIBM/ProdData/WebSphere/AppServer/V85/LibertyCore'>
</profile>
<rollback>
 <offering profile='WebSphere Liberty V8.5' id='com.ibm.websphere.liberty.v85' version='8.5.5.20101025_2108' />
</rollback>
</agent-input>
```

### Tips:

- The profile ID (`<profile . . . id='profile_ID' . . . .>` and `<offering . . . profile='profile_ID' . . . .>`) can be found when you run the `imcl listInstallationDirectories -verbose` command from the `eclipse/tools` subdirectory in the directory where you installed Installation Manager. It is the same as the package group's name.
- The offering ID (`<offering . . . id='offering_ID' . . . .>`) can be found in WebSphere Application Server product offerings for supported operating systems.
- The *version* is a specific version of the offering to which to rollback (8.5.5.20101025\_2108 for example).

This specification is optional.

- If *version* is **not** specified, the installation rolls back to the previously installed version of the offering and **all** interim fixes for that version are installed.
- If *version* is specified, the installation rolls back to the specified earlier version of the offering and **no** interim fixes for that version are installed.

- If you previously installed Liberty Repository features and addons but do not have access to a Liberty repository when you rollback your installation, specify the following properties in the response file:

```
<data key='user.feature' value='' />
<data key='user.addon' value='' />
```

Specifying these properties enables the product to rollback and uninstalls all features and addons.

## Using the sample response files

You can edit and use sample response files for installing, modifying, or uninstalling IBM Web Enablement Liberty for IBM i silently.

### Procedure

- “Sample response file: Installing IBM Web Enablement Liberty for IBM i” on page 821
- “Sample response file: Modifying IBM Web Enablement Liberty for IBM i” on page 825
- “Sample response file: Uninstalling IBM Web Enablement Liberty for IBM i” on page 830

## Sample response file: Installing IBM Web Enablement Liberty for IBM i:

You can edit and use this example of a response file for installing IBM Web Enablement Liberty for IBM i.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
Question 3. How do I store and pass credentials to repositories that
require authentication?
Answer 3. There are two methods for storing authentication credentials
for Installation Manager depending on the version being used,
either key ring files or storage files.
#
Versions of Installation Manger before 1.6.2 use a key ring file to store
encrypted credentials for authenticating with repositories. Follow this
two-step process for creating and using a key ring file with Installation Manager.
#
First, create a key ring file with your credentials by starting
Installation Manager from the command line under eclipse subdirectory
with the keyring parameter.
Use the optional password parameter to password protect your file.
#
Windows = IBMIM.exe -keyring <path and file name> -password <password>
Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
-password <password>
#
Installation Manager will start in graphical mode. Verify that the
repositories to which you need to authenticate are included in the
preferences, File / Preferences / Repositories. If they are not
listed, then click Add Repositories to add the URL or UNC path.
Installation Manager will prompt for your credentials. If the repository
is already in the list, then any attempt to access the repository location,
such as clicking the Test Connections button, will also prompt for your
credentials. Enter the correct credential and check the Save password
```

```

checkbox. The credentials are saved to the key ring file you specified.
#
Second, when you start a silent installation, run imcl under eclipse/tools
subdirectory, and provide Installation Manager with the location of the key
ring file and the password if the file is protected. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
#
Versions of Installation Manager at 1.6.2 or higher use storage files
to store encrypted credentials. Complete the following steps to create
master password and storage files to use with Installation Manager.
#
First, if you do not have a master password file already, create a text file
that contains a master passphrase. An example of a passphrase is:
"This text is the passphrase for a master password file."
#
Next, run imutilsc under the eclipse/tools subdirectory with the following
options to create and store user credentials in a storage file.
-secureStorageFile <path and file name of storage file>
-masterPasswordFile <path and file name of master password file>
-url <repository address> or -passportAdvantage <PPA repository address>
-userName <user name>
-userPassword <password for user>
#
Example of a command to create a storage file by operating system
Windows = imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile C:\IBM\credential.store
-masterPasswordFile C:\IBM\master_password_file.txt
Linux, UNIX, IBM z/OS, and the OS X operating system =
./imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile /home/IBM/credential.store
-masterPasswordFile /home/user/IBM/master_password_file.txt
#
Usage hints:
* Do not use both the -url and -passportAdvantage options in the same command.
* Enclose file paths that include spaces with double quotation marks.
* If you use the IBM service repositories, you can specify the value:
http://www.ibm.com/software/repositorymanager/entitled/repository.xml
for the -url option which is a generic service repository for IBM packages.
* Repeat steps to store credentials for multiple users in one file.
* Repeat steps to store credentials for multiple repositories in one file.
#
Afterwards, when you start a silent installation, run imcl under the eclipse/tools
subdirectory, and provide Installation Manager with the location of the storage
file. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
#
-->
<!-- ##### Agent Input #####
#
Note that the acceptLicense attribute has been deprecated.
Use the -acceptLicense command line option to accept license agreements.
#
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the installation finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->
<agent-input clean="true" temporary="true">
<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine

```

```

with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
 <!-- ##### IBM WebSphere Live Update Repositories #####
 # These repositories contain WebSphere Application Server Liberty offerings,
 # and updates for those offerings
 #
 # To use the secure repository (https), you must have an IBM ID,
 # which can be obtained by registering at: http://www.ibm.com/account
 # or your Passport Advantage account.
 #
 # And, you must use a key ring file with your response file.
 ##### -->
 <repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.WEBENAB.v85"/>
 <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

 <!-- ##### Custom Repositories #####
 # Uncomment and update the repository location key below
 # to specify URLs or UNC paths to any intranet repositories
 # and directory paths to local repositories to use.
 ##### -->
 <!-- <repository location='https:\\w3.mycompany.com\\repositories\\' /> -->
 <!-- <repository location='/home/user/repositories/websphere/' /> -->

 <!-- ##### Local Repositories #####
 # Uncomment and update the following line when using a local
 # repository located on your own machine to install a
 # WebSphere Application Server Liberty offering.
 ##### -->
 <!-- <repository location='insert the full directory path inside single quotes' /> -->
</server>

<!-- ##### Install Packages #####
#
Install Command
#
Use the install command to inform Installation Manager of the
installation packages to install.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be installed. The offering listed must be present in
at least one of the repositories listed earlier. The example
command below contains the offering ID for the WEBENAB
edition of WebSphere Application Server Liberty.
#
The version attribute is optional. If a version number is provided,
then the offering will be installed at the version level specified
as long as it is available in the repositories. If the version
attribute is not provided, then the default behavior is to install
the latest version available in the repositories. The version number
can be found in the repository.xml file in the repositories.
For example, <offering ... version='8.5.5000.20130326_0211'>.
#
The profile attribute is required and typically is unique to the
offering. If modifying or updating an existing installation, the
profile attribute must match the profile ID of the targeted installation
of WebSphere Application Server Liberty.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer,extprogmodels
#
The features embeddablecontainer,extprogmodels
are subfeatures of liberty.
#

```

```

You can use these functions to add or remove feature embeddablecontainer or extprogmodels later.
#
The installFixes attribute indicates whether fixes available in
repositories are installed with the product. By default, all
available fixes will be installed with the offering.
#
Valid values for installFixes:
none = do not install available fixes with the offering.
recommended = installs all available recommended fixes with the offering.
all = installs all available fixes with the offering.
#
Interim fixes for offerings also can be installed while they
are being installed by including the offering ID for the interim
fix and specifying the profile ID.
#
Installation Manager supports installing multiple offerings at once.
Additional offerings can be included in the install command,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=com.ibm.websphere.wlp.core.doc%2Fae%2Fwlp_ins_installation_dist_silent.html
#
-->
<install modify="false">
<offering id="com.ibm.websphere.liberty.WEBENAB.v85" profile="WebSphere Liberty V8.5" features="liberty,embeddablecontainer,extprogmodels" installFixes="none" />
</install>
<profile id="WebSphere Liberty V8.5" installLocation="/QIBM/ProdData/WebSphere/Liberty/V85/Express">
<data key="eclipseLocation" value="/QIBM/ProdData/WebSphere/Liberty/V85/Express" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location

```



```

false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false' />
-->

</agent-input>

```

## Sample response file: Modifying IBM Web Enablement Liberty for IBM i:

You can edit and use this example of a response file for modifying IBM Web Enablement Liberty for IBM i.

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?

```

```

Answer 2. Create a silent installation response file and run the following command
from the eclipse/tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
Question 3. How do I store and pass credentials to repositories that
require authentication?
Answer 3. There are two methods for storing authentication credentials
for Installation Manager depending on the version being used,
either key ring files or storage files.
#
Versions of Installation Manger before 1.6.2 use a key ring file to store
encrypted credentials for authenticating with repositories. Follow this
two-step process for creating and using a key ring file with Installation Manager.
#
First, create a key ring file with your credentials by starting
Installation Manager from the command line under eclipse subdirectory
with the keyring parameter.
Use the optional password parameter to password protect your file.
#
Windows = IBMIM.exe -keyring <path and file name> -password <password>
Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
-password <password>
#
Installation Manager will start in graphical mode. Verify that the
repositories to which you need to authenticate are included in the
preferences, File / Preferences / Repositories. If they are not
listed, then click Add Repositories to add the URL or UNC path.
Installation Manager will prompt for your credentials. If the repository
is already in the list, then any attempt to access the repository location,
such as clicking the Test Connections button, will also prompt for your
credentials. Enter the correct credential and check the Save password
checkbox. The credentials are saved to the key ring file you specified.
#
Second, when you start a silent installation, run imcl under eclipse/tools
subdirectory, and provide Installation Manager with the location of the key
ring file and the password if the file is protected. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-keyring <path and name of key ring file> -password <password>
#
Versions of Installation Manager at 1.6.2 or higher use storage files
to store encrypted credentials. Complete the following steps to create
master password and storage files to use with Installation Manager.
#
First, if you do not have a master password file already, create a text file
that contains a master passphrase. An example of a passphrase is:
"This text is the passphrase for a master password file."
#
Next, run imutilsc under the eclipse/tools subdirectory with the following
options to create and store user credentials in a storage file.
-secureStorageFile <path and file name of storage file>
-masterPasswordFile <path and file name of master password file>
-url <repository address> or -passportAdvantage <PPA repository address>
-userName <user name>
-userPassword <password for user>
#
Example of a command to create a storage file by operating system
Windows = imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile C:\IBM\credential.store
-masterPasswordFile C:\IBM\master_password_file.txt
Linux, UNIX, IBM z/OS, and the OS X operating system =
./imutilsc saveCredential -url http://myServer -userName myUserName
-userPassword myPassword -secureStorageFile /home/IBM/credential.store
-masterPasswordFile /home/user/IBM/master_password_file.txt
#
Usage hints:
* Do not use both the -url and -passportAdvantage options in the same command.
* Enclose file paths that include spaces with double quotation marks.
* If you use the IBM service repositories, you can specify the value:
http://www.ibm.com/software/repositorymanager/entitled/repository.xml

```

```

for the -url option which is a generic service repository for IBM packages.
* Repeat steps to store credentials for multiple users in one file.
* Repeat steps to store credentials for multiple repositories in one file.
#
Afterwards, when you start a silent installation, run imcl under the eclipse/tools
subdirectory, and provide Installation Manager with the location of the storage
file. For example:
#
Windows = imcl.exe -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <path and file name of response file>
-secureStorageFile <path and name of storage file>
-masterPasswordFile <path and name of master password file>
#
-->

<!-- ##### Agent Input #####
#
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the installation finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
 <!-- ##### IBM WebSphere Live Update Repositories #####
 # These repositories contain WebSphere Application Server Liberty offerings,
 # and updates for those offerings
 #
 # To use the secure repository (https), you must have an IBM ID,
 # which can be obtained by registering at: http://www.ibm.com/account
 # or your Passport Advantage account.
 #
 # And, you must use a key ring file with your response file.
 ##### -->
 <repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.WEBENAB.v85"/>
 <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

 <!-- ##### Custom Repositories #####
 # Uncomment and update the repository location key below
 # to specify URLs or UNC paths to any intranet repositories
 # and directory paths to local repositories to use.
 ##### -->
 <!-- <repository location="https://w3.mycompany.com/repositories"/> -->
 <!-- <repository location="/home/user/repositories/websphere"/> -->

 <!-- ##### Local Repositories #####
 # Uncomment and update the following line when using a local
 # repository located on your own machine to install a
 # WebSphere Application Server Liberty offering.
 ##### -->
 <!-- <repository location="insert the full directory path inside single quotes"/> -->
</server>

<!-- ##### Modify Packages #####
#

```

```

Install and Uninstall Commands
#
Use the install and uninstall commands to inform Installation Manager
of the installation packages to install or uninstall.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be installed. The offering listed must be present in
at least one of the repositories listed earlier. The example
command below contains the offering ID for the WEBENAB
edition of WebSphere Application Server Liberty.
#
The version attribute is optional. If a version number is provided,
then the offering will be installed or uninstalled at the version level
specified as long as it is available in the repositories. If the version
attribute is not provided, then the default behavior is to install or
uninstall the latest version available in the repositories. The version
number can be found in the repository.xml file in the repositories.
For example, <offering ... version='8.5.5000.20130328_1111'>.
#
The profile attribute is required and typically is unique to the
offering. If modifying or updating an existing installation, the
profile attribute must match the profile ID of the targeted installation
of WebSphere Application Server Liberty.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer,extprogmodels
#
The features embeddablecontainer,extprogmodels
are subfeatures of liberty.
#
You can use these functions to add or remove feature embeddablecontainer
or extprogmodels later.
#
In the example that follows, the feature embeddablecontainer and extprogmodels
are being added and no feature are being removed from the specified offering.
#
The installFixes attribute indicates whether fixes available in
repositories are installed with the product. By default, all
available fixes will be installed with the offering.
#
Valid values for installFixes:
none = do not install available fixes with the offering.
recommended = installs all available recommended fixes with the offering.
all = installs all available fixes with the offering.
#
Installation Manager supports modifying multiple offerings at once.
Additional offerings can be included in the install and uninstall commands,
with each offering requiring its own offering ID, version, profile value,
and feature values.
#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Ftwlp_ins_installation_dist_silent.html
#
-->

```

```

<install modify="true">
<offering id="com.ibm.websphere.liberty.WEBENAB.v85" profile="WebSphere Liberty V8.5" features="embeddablecontainer,extprogmodels" />
</install>
<profile id="WebSphere Liberty V8.5" installLocation="/QIBM/ProdData/WebSphere/Liberty/V85/Express">
<data key="eclipseLocation" value="/QIBM/ProdData/WebSphere/Liberty/V85/Express" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of
Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30' />
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45' />
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0' />
<preference name='offering.service.repositories.areUsed' value='true' />
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false' />
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false' />
<preference name='http.ntlm.auth.kind' value='NTLM' />
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true' />
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false' />
<preference name='PassportAdvantageIsEnabled' value='false' />
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false' />
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false' />
-->

</agent-input>

```

## Sample response file: Uninstalling IBM Web Enablement Liberty for IBM i:

You can edit and use this example of a response file for uninstalling IBM Web Enablement Liberty for IBM i.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #####
Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2013.
All Rights Reserved. US Government Users Restricted Rights-Use, duplication
or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->

<!-- ##### Frequently Asked Questions #####
The latest information about using Installation Manager is
located in the online Information Center. There you can find
information about the commands and attributes used in
silent installation response files.
#
Installation Manager Information Center can be found at:
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
Question 1. How do I record a response file using Installation Manager?
Answer 1. Start Installation Manager from the command line under the
eclipse subdirectory with the record parameter and it will generate a
response file containing actions it performed, repositories it used, and
its preferences settings. Optionally use the -skipInstall parameter if
you do not want the product to be installed to the machine. Specify a
new agentDataLocation location value when doing a new installation. Do
not use an existing agentDataLocation for an installation because it might
damage the installation data and prevent you from modifying, updating,
rolling back, or uninstalling the installed packages.
#
Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
For example:
Windows = IBMIM.exe -record c:\temp\responsefiles\WASv85.install.Win32.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv85.install.RHEL64.xml
-skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
Question 2. How do I run Installation Manager silently using response file?
Answer 2. Create a silent installation response file and run the following command
from the eclipse\tools subdirectory in the directory where you installed
Installation Manager:
#
Windows = imcl.exe -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input <response_file_path_and_name> -log <log_file_path_and_name>
#
For example:
Windows = imcl.exe -acceptLicense -showProgress
input c:\temp\responsefile\WASv85.install.Win32.xml
Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
input /home/user/responsefile/WASv85.install.RHEL64.xml
#
The -acceptLicense command must be included to indicate acceptance of all
license agreements of all offerings being installed, updated or modified.
The -showProgress command shows progress when running in silent mode.
Additional commands can be displayed by requesting help: IBMIM -help
#
-->

<!-- ##### Agent Input #####
The clean and temporary attributes specify the repositories and other
preferences Installation Manager uses and whether those settings
should persist after the uninstall finishes.
#
Valid values for clean:
true = only use the repositories and other preferences that are
specified in the response file.
false = use the repositories and other preferences that are
specified in the response file and Installation Manager.
#
Valid values for temporary:
true = repositories and other preferences specified in the
response file do not persist in Installation Manager.
false = repositories and other preferences specified in the
response file persist in Installation Manager.
#
-->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #####
Repositories are locations that Installation Manager queries for
```

```

installable packages. Repositories can be local (on the machine
with Installation Manager) or remote (on a corporate intranet or
hosted elsewhere on the internet).
#
If the machine using this response file has access to the internet,
then include the IBM WebSphere Live Update Repositories in the list
of repository locations.
#
If the machine using this response file cannot access the internet,
then comment out the IBM WebSphere Live Update Repositories and
specify the URL or UNC path to custom intranet repositories and
directory paths to local repositories to use.
#
-->

<server>
<!-- ##### IBM WebSphere Live Update Repositories #####
These repositories contain WebSphere Application Server Liberty offerings,
and updates for those offerings
#
To use the secure repository (https), you must have an IBM ID,
which can be obtained by registering at: http://www.ibm.com/account
or your Passport Advantage account.
#
And, you must use a key ring file with your response file.
-->
<!-- repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.WEBENAB.v85"/-->
<!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

<!-- ##### Custom Repositories #####
Uncomment and update the repository location key below
to specify URLs or UNC paths to any intranet repositories
and directory paths to local repositories to use.
-->
<!-- <repository location='https:\w3.mycompany.com\repositories\' /> -->
<!-- <repository location='/home/user/repositories/websphere/' /> -->

<!-- ##### Local Repositories #####
Uncomment and update the following line when using a local
repository located on your own machine to install a
WebSphere Application Server Liberty offering.
-->
<!-- <repository location='insert the full directory path inside single quotes' /> -->
</server>

<!-- ##### Uninstall Packages #####
#
Uninstall Command
#
Use the uninstall command to inform Installation Manager of the
installation packages to uninstall.
#
The modify attribute is optional and can be paired with an install
command to add features or paired with an uninstall command to
remove commands. If omitted, the default value is set to false.
false = indicates not to modify an existing install by adding
or removing features.
true = indicates to modify an existing install by adding or
removing features.
#
The offering ID attribute is required because it specifies the
offering to be uninstalled. The example command below contains the
offering ID for WebSphere Application Server Liberty WEBENAB edition.
#
The version attribute is optional. If a version number is provided,
then the offering will be uninstalled at the version level specified
If the version attribute is not provided, then the default behavior is
to uninstall the latest version. The version number can be found in
the repository.xml file in the repositories.
For example, <offering ... version='8.5.5000.20130326_0211'>.
#
The profile attribute is required and must match the package group
name for the offering to be uninstalled.
#
The features attribute is optional. Offerings always have at least
one feature; a required core feature which is installed regardless
of whether it is explicitly specified. If other feature names
are provided, then only those features will be installed.
Features must be comma delimited without spaces.
#
The feature values for WebSphere Application Server Liberty include:
liberty,embeddablecontainer,extprogmodels
#
The features embeddablecontainer,extprogmodels
are subfeatures of liberty.
#
Installation Manager supports uninstalling multiple offerings at once.
Additional offerings can be included in the uninstall command,
with each offering requiring its own offering ID, version, profile value,
and feature values.

```

```

#
Profile Command
#
A separate profile command must be included for each offering listed
in the install command. The profile command informs Installation
Manager about offering specific properties or configuration values.
#
The installLocation specifies where the offering will be installed.
If the response file is used to modify or update an existing
installation, then ensure the installLocation points to the
location where the offering was installed previously.
#
The eclipseLocation data key should use the same directory path to
WebSphere Application Server Liberty as the installationLocation attribute.
#
Include data keys for product specific profile properties.
For instance, Installing WebSphere Application Server Liberty Offerings on
a 64-bit system will require to include one of the options for an IBM Software
Development Kit, this can be specified by data key cic.selector.arch, its value
can be either x86 (for 32-bit), or x86_64 (for 64-bit).
#
More details for cic.selector.arch can be found in the link below:
#
https://infocenters.hursley.ibm.com/was/vNext/draft/help/index.jsp?topic=%2Fcom.ibm.websphere.wlp.core.doc%2Fae%2Ftwlp_ins_installation_dist_silent.html
#
-->

<uninstall modify="false">
<offering id="com.ibm.websphere.liberty.WEBENAB.v85" profile="WebSphere Liberty V8.5" features="liberty,embeddablecontainer,extprogmodels" />
</uninstall>
<profile id="WebSphere Liberty V8.5" installLocation="/QIBM/ProdData/WebSphere/Liberty/V85/Express">
<data key="eclipseLocation" value="/QIBM/ProdData/WebSphere/Liberty/V85/Express" />
<data key="cic.selector.arch" value="x86_64" />
</profile>

<!-- ##### Shared Data Location #####
Uncomment the preference for eclipseCache to set the shared data
location the first time you use Installation Manager to do an
installation.
#
Eclipse cache location can be obtained from the installed.xml file found in
Linux/Unix: /var/ibm/InstallationManager
Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
from the following property:
<property name='cacheLocation' value='C:\Program Files\IBM\IMShared' />
#
Open the installed.xml file in a text editor because the style sheet
might hide this value if opened in a web browser.
For further information on how to edit preferences, refer to the public library at:
http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
After the shared data location is set, it cannot be changed
using a response file or the graphical wizard.
#
Ensure that the shared data location is a location that can be written
to by all user accounts that are expected to use Installation Manager.
#
By default, Installation Manager saves downloaded artifacts to
the shared data location. This serves two purposes.
#
First, if the same product is installed a more than once to the machine,
then the files in the shared data location will be used rather than
downloading them again.
#
Second, during the rollback process, the saved artifacts are used.
Otherwise, if the artifacts are not saved or are removed, then
Installation Manager must have to access the repositories used to
install the previous versions.
#
Valid values for preserveDownloadedArtifacts:
true = store downloaded artifacts in the shared data location
false = remove downloaded artifacts from the shared data location
#
-->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared' />
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true' />
-->

<!-- ##### Preferences Settings #####
Additional preferences for Installation Manager can be specified.
These preference correspond to those that are located in the graphical
interface under File / Preferences.
#
If a preference command is omitted from or commented out of the response
file, then Installation Manager uses the preference value that was
previously set or the default value for the preference.
#
Preference settings might be added or deprecated in new versions of

```



```

Installation Manager. Consult the online Installation Manager
Information Center for the latest set of preferences and
descriptions about how to use them.
#
http://publib.boulder.ibm.com/infocenter/install/v1r6/index.jsp
#
-->
<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45'/>
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false'/>
-->
</agent-input>

```

## Installing and uninstalling Liberty using downloaded files and archives

### Distributed operating systems

IBM i

You can install Liberty by extracting an archive file.

### Distributed operating systems

IBM i

## About this task

To try out Liberty and use Liberty to develop applications that run on WebSphere Application Server traditional or Liberty, you can download a no-charge, unsupported edition from the WASdev download page.

To use Liberty in a production environment with guaranteed service levels and IBM support, you must purchase WebSphere Application Server (base), WebSphere Application Server Network Deployment, WebSphere Application Server Express, or WebSphere Application Server Liberty Core. Liberty is included with these editions and can also be downloaded separately, as an edition-specific Java archive (JAR) file, from Passport Advantage Online. The associated service is available from Fix Central. If you download and install Liberty from an unsupported JAR or ZIP file, you can later purchase a supported edition and upgrade the license for your existing installation.

The following table lists where you can download each type of installation archive file.

Table 63. Installation archive file locations

File	Content	Passport Advantage Online	IBM Fix Central	WASdev website
JAR files for installing a supported edition-specific V8.5.5.8 or later Liberty runtime environment  Example: wlp-nd-all-8.5.5.x.jar	<b>Runtime</b> Edition-specific, V8.5.5.8 or later  <b>Features</b> All features that apply to that edition		⏏	

Table 63. Installation archive file locations (continued)

File	Content	Passport Advantage Online	IBM Fix Central	WASdev website
<p>ZIP files for installing an unsupported V8.5.5.6 or later Liberty runtime environment with a limited production license</p> <p>Can be upgraded to a supported edition by using the license upgrade JAR file</p> <p>Example: wlp-webProfile7-8.5.5.x.zip</p>	<p><b>Runtime</b> V8.5.5.6 or later</p> <p><b>Features</b> Optional Java EE 7 feature sets</p> <p><b>SDK</b> Optional IBM Java 8 SDK</p>		△	△
<p>JAR files for installing an edition-specific V8.5.5.0 Liberty runtime environment</p> <p>Example: wlp-nd-runtime-8.5.5.0.jar</p>	<p><b>Runtime</b> Edition-specific, V8.5.5.0</p> <p><b>Features</b> V8.5.5.0 features</p>	△		
<p>JAR files for installing an edition-specific V8.5.5.1 or later Liberty runtime environment</p> <p>The unsupported JAR file can be upgraded to a supported edition by using the license upgrade JAR file</p> <p>Example: wlp-nd-runtime-8.5.5.6.jar</p>	<p><b>Runtime</b> Edition-specific, V8.5.5.1 or later</p> <p><b>Features</b> Up to the latest levels of the V8.5.5.2 feature set</p>		Fully supported JAR files	Unsupported JAR file only
<p>JAR files for license upgrades:</p> <ul style="list-style-type: none"> <li>Upgrade trial or unsupported editions to supported editions</li> <li>Upgrade supported editions to other supported editions with more functionality</li> </ul> <p>Example: wlp-nd-license-8.5.5.jar</p>	<p>No runtime environment or features</p>	△	△	

**Distributed operating systems** For Windows, Linux, and Mac OS, you can also install the WebSphere Application Server Developer Tools for Eclipse, then use the tools to install the profile. For more information, see “Installing Liberty developer tools and (optionally) Liberty” on page 835.

**Distributed operating systems** IBM i

## Procedure

- **8.5.5.8** You can install Liberty and all features that apply to the edition from a self-extracting JAR file.
  1. Install Liberty from the JAR file.
  2. Optional: Upgrade your product edition or license. For example, you can upgrade from WebSphere Application Server Liberty - Express to WebSphere Application Server Liberty Network Deployment for access to advanced functions.
  3. Update your Liberty installation as needed.
- **8.5.5.6** You can install Liberty and optional features in a single step by extracting an installation ZIP file. Installing Liberty from the ZIP files enables no-charge, unsupported, unlimited use of Liberty in development environments and unsupported limited use in small-scale test and production environments.
  1. Review the available installation ZIP files. You can install only the Liberty kernel with no additional features, or you can install Liberty and a set of features that support application development in certain contexts, such as Java Platform, Enterprise Edition 7.

If you want to install Liberty and IBM SDK Java Technology Edition Version 8, choose the `wlp-webProfile7-java8-<platform>-<architecture>-<version>.zip` that fits your system.
  2. Install Liberty from the ZIP file. After you install Liberty, you have everything that you need to get started with your chosen set of features.
  3. Optional: Install additional Liberty Repository assets by using the **installUtility** command. You can also uninstall any unwanted features.
  4. Optional: Upgrade your Liberty profile to an unlimited production edition. Upgrading your license gives you full access to IBM support in an unlimited test or production environment.
  5. Update your Liberty installation as needed.
- You can install Liberty from a self-extracting JAR file. The JAR files contain the Version 8.5.5.2 feature set, but you can install additional features after you install Liberty.
  1. Install Liberty from the JAR file.
  2. Install Liberty Repository assets.
  3. Optional: Upgrade your product edition or license. For example, you can upgrade from WebSphere Application Server Liberty - Express to WebSphere Application Server Liberty Network Deployment for access to advanced functions.
  4. Update your Liberty installation as needed.

## Installing Liberty developer tools and (optionally) Liberty

### Distributed operating systems

The developer tools for Liberty are available as part of the IBM WebSphere Application Server Developer Tools for Eclipse. In addition, WebSphere Application Server Developer Tools for Eclipse is a lightweight set of tools for developing, assembling, and deploying Java EE, OSGi, Web 2.0, and mobile applications to WebSphere Application Server.

### About this task

To install WebSphere Application Server Developer Tools for Eclipse, see Installing WebSphere Application Server Developer Tools for Eclipse.

## What to do next

After you install the developer tools, you can optionally install WebSphere Application Server Liberty. The developer tools can install Liberty for you when you create a server for the first time. For more information about installing Liberty by using the developer tools, see “Creating a Liberty server by using developer tools” on page 884.

To learn more about Liberty Core, see Chapter 1, “WebSphere Application Server Liberty Core: Overview,” on page 1.

## Installing Liberty by extracting a Java archive file

Distributed operating systems

IBM i

By running a self-extracting Java archive (JAR) file that contains the distribution image, you can install Liberty and you are ready to create a server.

### Before you begin

Your system must meet the operating system and Java requirements for using Liberty. See System Requirements for WebSphere Application Server V8.5.5.

IBM i

For the IBM i platform, this topic assumes that the minimum supported Java level is installed at one of the following locations:

- /QOpenSys/QIBM/ProdData/JavaVM/jdk626/32bit
- /QOpenSys/QIBM/ProdData/JavaVM/jdk626/64bit

### Procedure

1. Get a copy of the distribution image JAR file.
  - **8.5.5.8** You can download a single, edition-specific archive file that contains the Liberty runtime environment with all applicable features from IBM Fix Central.  
These JAR files are called *wlp-edition-all-version.jar*, such as *wlp-core-all-8.5.5.8.jar* or *wlp-nd-all-8.5.5.8.jar*.
  - You can download separately packaged archive files that contain the Liberty runtime with the Version 8.5.5.2 feature set.  
The edition-specific runtime JAR files are called *wlp-edition-runtime-version.jar*.
    - Download the archive file for the no-charge, unsupported edition from the WASdev download page. If you install from this JAR file, you can upgrade later to a supported edition.
    - Download the archive file for the initial release, which includes IBM support, from Passport Advantage online. The associated fix packs are available from Fix Central.

For a list of all available archives, see “List of installation Java archive files” on page 838.
2. Extract the distribution images to your preferred directory.  
To extract the distribution image by using the interactive install wizard, run `java -jar wlp-archive-name.jar`. All application server files are stored in subdirectories of the *wlp* directory.  
For a list of the available extraction options, see “Java archive file extraction options” on page 837.
3. Optional: Set the **JAVA\_HOME** property for your environment.  
Liberty runs in a Java Runtime Environment (JRE). It does not share the JDK or JRE that WebSphere Application Server traditional uses. You can specify the JDK or JRE location using the **JAVA\_HOME** property in the `server.env` file, as described in “Customizing the Liberty environment” on page 947. When you set the **JAVA\_HOME** property in the `server.env` file, Liberty uses the same Java runtime location regardless of the user profile that Liberty server runs under.

**IBM i** On the IBM i platform, setting the **JAVA\_HOME** property as a system-level environment variable is not recommended. The IBM i platform is a shared environment, and changing system-level environment variables might affect other applications.

**Distributed operating systems** On Linux or UNIX systems, you can instead set **JAVA\_HOME** in the user `.bashrc` file, or append the JDK or JRE path to the **PATH** environment variable. On Windows systems, you can instead set **JAVA\_HOME** as a system environment variable, or append the JDK or JRE path to the **PATH** system variable. For example, on Windows systems you can use the following commands to set the **JAVA\_HOME** property, and to add the Java `/bin` directory to the path:

```
set JAVA_HOME=C:\Progra~1\Java\JDK16
set PATH=%JAVA_HOME%\bin;%PATH%
```

**Note:** **8.5.5.11** Support for using Java SE 6 with WebSphere Liberty ends in September 2017. After the end of support, the Liberty kernel will be recompiled and can no longer run with Java SE 6. If you continue to use Java SE 6 on earlier fix packs after the end of support date, you could expose your environment to security risks.

Java SE 8 is the recommended Java SDK because it provides the latest features and security updates. You can install it by installing the IBM SDK, Java Technology Edition, Version 8 package to the package group that contains WebSphere Liberty.

**Note:** The Liberty runtime environment searches for the **java** command in this order: **JAVA\_HOME** property, **JRE\_HOME** property, and system **PATH** property.

For more information about supported Java environments, and where to get them, see “Minimum supported Java levels” on page 1481.

4. Optional: Upgrade your Liberty installation to a more advanced supported edition. For example, you can upgrade from WebSphere Application Server Liberty Core to WebSphere Application Server Network Deployment. For more information, see “Upgrading Liberty installations” on page 847.

## What to do next

After you install Liberty, you can further customize your environment by installing additional assets; see “Installing Liberty Repository assets” on page 852.

**IBM i** On the IBM i platform, after you extract the distribution image, you can configure all servers to run as jobs in the batch subsystem under the QEJBSVR user profile that is provided with the product. For more information, see “Configuring the Liberty server to start as a job in the QWAS85 subsystem on IBM i” on page 850.

## Java archive file extraction options

You can install Liberty by extracting a Java archive (JAR) file. By running a self-extracting JAR file that contains the distribution image, you can install Liberty and you are ready to create a server.

### Syntax

The command syntax is as follows:

```
java -jar archive-file-name.jar [Options] [install location]
```

where *archive-file-name.jar* is the name of the archive file you are extracting.

### Note:

- If you do not specify any options, the distribution image is extracted by using the interactive installer.
- If you do not specify an installation location, the default target directory is the location of the archive file.

## Options

The following options are available for the extraction command:

**--help** Display a short explanation about how to use the command and a list of available options

**--acceptLicense**

Automatically indicate acceptance of license terms and conditions

**--verbose**

Display detailed information during archive extraction

**--viewLicenseAgreement**

View license agreement

**--viewLicenseInfo**

View license information

## Usage

The following examples demonstrate the correct syntax:

```
java -jar myarchivefile.jar
java -jar myarchivefile.jar --help
java -jar myarchivefile.jar --acceptLicense C:\Liberty-install
java -jar myarchivefile.jar --verbose
java -jar myarchivefile.jar --viewLicenseAgreement
java -jar myarchivefile.jar --viewLicenseInfo
```

## List of installation Java archive files

You can install Liberty by extracting a Java archive (JAR) file.

**Note:** 8.5.5.6 You can alternatively install Liberty and optional features from the installation ZIP files. For more information, see “Installing Liberty by extracting a ZIP archive file” on page 842.

8.5.5.8 Java archive files that contain all applicable features

The following JAR files contain the Liberty runtime and all features that apply to the specific edition.

### WebSphere Application Server Network Deployment

wlp-nd-all-<version>.jar

The WebSphere Application Server Network Deployment Liberty runtime environment and all Liberty features that apply to the edition.

### WebSphere Application Server (base)

wlp-base-all-<version>.jar

The single-server WebSphere Application Server Liberty runtime environment and all Liberty features that apply to the edition.

### WebSphere Application Server Liberty Core

wlp-core-all-<version>.jar

The WebSphere Application Server Liberty Core runtime environment and all Liberty features that apply to the edition.

## Java archive files that contain the Version 8.5.5.2 feature set

The following JAR files contain up to the Version 8.5.5.2 feature set at the latest level of code for the fix pack that you install. You can install additional assets from the Liberty Repository. For more information, see “Installing Liberty Repository assets” on page 852.

## WebSphere Application Server Liberty Core

### wlp-core-runtime-*<version>*.jar

The full WebSphere Application Server Liberty Core installation.

### wlp-extras-*<version>*.jar

The embeddable EJB Container, and JPA client, to help with developing and testing EJB Lite and JPA applications. You must install this JAR file in an empty directory.

## WebSphere Application Server Liberty Core trial

### wlp-core-trial-runtime-*<version>*.jar

The full WebSphere Application Server Liberty Core installation. licensed for trial use.

## Features included in the JAR files

The following table compares the features that are included in the JAR files that contain the Version 8.5.5.2 feature set. For more information about these features, see “Liberty features” on page 483.

Table 64. Features included in the JAR files

Feature	WebSphere Application Server Liberty Core	WebSphere Application Server (base)	WebSphere Application Server Express	WebSphere Application Server Network Deployment	WebSphere Application Server for Developers (IPLA and ILAN)
<b>Runtime JAR files</b>	wlp-core-runtime- <i>&lt;version&gt;</i> .jar wlp-core-trial-runtime- <i>&lt;version&gt;</i> .jar	wlp-core-runtime- <i>&lt;version&gt;</i> .jar wlp-base-server-submodule.jar wlp-runtime- <i>&lt;version&gt;</i> .jar	wlp-express-runtime- <i>&lt;version&gt;</i> .jar wlp-expression.jar	wlp-network-deployment- <i>&lt;version&gt;</i> .jar	wlp-developers-runtime-ipla- <i>&lt;version&gt;</i> .jar wlp-developers-runtime- <i>&lt;version&gt;</i> .jar
<b>Java EE 6 Web Profile</b>					
beanValidation-1.0 cdi-1.0 ejbLite-3.1 jaxrs-1.1 jdbc-4.0 jndi-1.0 jpa-2.0 jsf-2.0 json-1.0 jsp-2.2 managedBeans-1.0 servlet-3.0 webProfile-6.0	⊆	⊆	⊆	⊆	⊆
<b>Enterprise OSGi</b>					
blueprint-1.0 osgi.jpa-1.0 wab-1.0	⊆	⊆	⊆	⊆	⊆
<b>Operations</b>					

Table 64. Features included in the JAR files (continued)

Feature	WebSphere Application Server Liberty Core	WebSphere Application Server (base)	WebSphere Application Server Express	WebSphere Application Server Network Deployment	WebSphere Application Server for Developers (IPLA and ILAN)
appSecurity-1.0 appSecurity-2.0 distributedMap-1.0 ldapRegistry-3.0 localConnector-1.0 monitor-1.0 oauth-2.0 osgiConsole-1.0 restConnector-1.0 serverStatus-1.0 sessionDatabase-1.0 ssl-1.0 timedOperations-1.0 webCache-1.0					
<b>Systems Management</b>					
clusterMember-1.0 collectiveController-1.0					
collectiveMember-1.0 concurrent-1.0					
<b>Extended Programming Model JAR files</b>	N/A	wlp-extended- <i>&lt;version&gt;</i>	wlp-extended- <i>&lt;version&gt;</i>	wlp-extended- <i>&lt;version&gt;</i>	wlp-extended- <i>&lt;version&gt;</i> .jar wlp-developers-extended- <i>&lt;version&gt;</i>
jaxb-2.2 jaxws-2.2 jmsMdb-3.1 mongodb-2.0 wasJmsClient-1.1 wasJmsSecurity-1.0 wasJmsServer-1.0 wmqJmsClient-1.1 wsSecurity-1.1					

## Applying a fix pack to a Liberty Java archive installation

Distributed operating systems

IBM i

Liberty offers a self-extracting Java archive-based installation as an alternative to using IBM Installation Manager. If you installed Liberty by using the self-extracting archive, and want to upgrade to the latest fix pack version, you can apply a new fix pack archive to a new location, then move any required user files and server configuration data.

### About this task

If you used IBM Installation Manager to install Liberty, you must use Installation Manager to apply a fix pack.

**Important:** You must apply a new fix pack archive to a new location.



## Procedure

1. Install the new runtime environment.
  - a. Copy or download the new fix pack archive onto the target system.
  - b. Launch the archive by using a Java command. You must use a Java command because the archive is an executable JAR file. Run the following command:
    - `java -jar <downloaded_archive_location>/<downloaded_archive_file_name>`For more information about using a Java command to launch an archive, see the instructions in “Installing Liberty by extracting a Java archive file” on page 836.
  - c. Review the license terms, and accept them to continue with the installation.
  - d. Select the installation location. Use a different location to where the previous version is installed.
2. Move any user data and server configurations. Liberty defines two locations for storing user-generated content and server configurations:

- WLP\_USER\_DIR; The location of server configuration files, including shared resources.
- WLP\_OUTPUT\_DIR; The location of resources generated by the server. For example, log files and temporary disk storage.

If the WLP\_USER\_DIR environment variable has been set on your system, then the new runtime environment will continue to use the same location. This results in no backup of server configuration data. To ensure that your server configuration is backed up, copy the directory referenced by WLP\_USER\_DIR to a new location on your file system. To protect the original environment, change the value of WLP\_USER\_DIR to point to the new location. During uninstallation, reset the value of WLP\_USER\_DIR to the location of the original server configuration.

If WLP\_USER\_DIR has not been set, the server configuration and shared resources are stored in the `usr` directory at the root of the server's runtime environment (for example, `<liberty_server_runtime_root>/usr`). During uninstallation of the runtime environment, you can reset the WLP\_USER\_DIR environment variable.

If the WLP\_OUTPUT\_DIR environment variable is set on your system, the new server also uses this location. This can result in old log files being overwritten. To ensure that old log files are protected, either update or unset the WLP\_OUTPUT\_DIR environment variable. During uninstallation, reset this value to its original value.

If the WLP\_OUTPUT\_DIR value is not set, the default location is in the server root directory (for example `<liberty_server_runtime_root>/usr/servers/<serverName>`). If the new runtime environment is installed to a new location, no updates are required during installation or uninstallation because logs continue to appear under the `usr/servers/<serverName>/logs` directory of each respective installation.

**Note:** If the `server.xml` file, or any included XML configuration file, references another resource outside the server configuration directory, these resources must also be copied across, or the references will need to be updated. This also applies to any resources that the application references directly, such as references to hardcoded paths on file systems. During uninstallation of the fix pack, these values can be manually reset to their original values.

3. Start the new server. Run `<liberty_VX+>/bin/server start <server_name>`.

## Removing a fix pack from a Liberty Java archive installation:

Distributed operating systems

IBM i

If you installed Liberty by using the self-extracting Java archive, you can uninstall a fix pack from the Liberty runtime environment in a given location by migrating any user data and server configurations to the previous Liberty profile runtime environment in a different location, and deleting the fix pack runtime environment.

## Procedure

1. Stop all servers running on the system.
  - `<liberty_VX>/bin/server stop <server_name>`

2. Move any user data and server configurations. For more information see the Move any user data and server configurations step in the installation task.
3. Delete the fix pack runtime environment.
4. Start the servers.
  - `<liberty_VX->/bin/server start <server_name>`

## Installing Liberty by extracting a ZIP archive file

Distributed operating systems

IBM i

8.5.5.6

You can install Liberty and optional features by extracting a ZIP archive file. These ZIP files are designed to help you quickly get started with Liberty.

### Before you begin

Your system must meet the operating system and Java requirements for using Liberty. See System Requirements for WebSphere Application Server V8.5.5.

**IBM i** For the IBM i platform, this topic assumes that the minimum supported Java level is installed at one of the following locations:

- /QOpenSys/QIBM/ProdData/JavaVM/jdk626/32bit
- /QOpenSys/QIBM/ProdData/JavaVM/jdk626/64bit


### About this task

Installing Liberty from the ZIP files enables no-charge, unsupported, unlimited use of Liberty in development environments and limited use in small-scale test and production environments. For more information, see the license information and license agreement files in the `wlp\lfiles` directory or WASdev.net. For access to IBM support and unlimited test and production use, you can later purchase and upgrade to a supported edition from your existing installation.

You can choose from several ZIP files to fit your needs:

- Java Platform, Enterprise Edition (Java EE) 7 technologies: Install the Liberty runtime environment plus features that support the Java EE 7 full platform, Web Profile, or application client
- IBM Java 8: Install the Liberty runtime environment, features that support the Java EE 7 Web Profile, and IBM SDK Java Technology Edition Version 8
- **8.5.5.9** OSGi technologies: Install the Liberty runtime environment and features that support the Java EE 7 Web Profile and OSGi
- Liberty kernel only: Install only the Liberty runtime environment

For a list of the available ZIP archive files, see “List of installation ZIP archive files” on page 843.

 **Watch:** Video: Installing Liberty from a ZIP file shows how you can quickly install Liberty from a ZIP archive file, start the server and add applications, and upgrade to a supported installation [Transcript]

### Procedure

1. Download the ZIP file of the distribution image from Fix Central or the product asset page on WASdev.net. The `wlp-<name>-<version>.zip` files contain logical groupings of features.
2. Extract the distribution image to your preferred directory. All of the application server files are stored in subdirectories of the `wlp` directory.
3. Optional: Set the **JAVA\_HOME** property for your environment.  
Liberty runs in a Java Runtime Environment (JRE). It does not share the JDK or JRE that WebSphere Application Server traditional uses. You can specify the JDK or JRE location using the **JAVA\_HOME**

property in the `server.env` file, as described in “Customizing the Liberty environment” on page 947. When you set the `JAVA_HOME` property in the `server.env` file, Liberty uses the same Java runtime location regardless of the user profile that Liberty server runs under.

**IBM i** On the IBM i platform, setting the `JAVA_HOME` property as a system-level environment variable is not recommended. The IBM i platform is a shared environment, and changing system-level environment variables might affect other applications.

**Distributed operating systems** On Linux or UNIX systems, you can instead set `JAVA_HOME` in the user `.bashrc` file, or append the JDK or JRE path to the `PATH` environment variable. On Windows systems, you can instead set `JAVA_HOME` as a system environment variable, or append the JDK or JRE path to the `PATH` system variable. For example, on Windows systems you can use the following commands to set the `JAVA_HOME` property, and to add the Java `/bin` directory to the path:

```
set JAVA_HOME=C:\Progra~1\Java\JDK16
set PATH=%JAVA_HOME%\bin;%PATH%
```

**Note:** **8.5.5.11** Support for using Java SE 6 with WebSphere Liberty ends in September 2017. After the end of support, the Liberty kernel will be recompiled and can no longer run with Java SE 6. If you continue to use Java SE 6 on earlier fix packs after the end of support date, you could expose your environment to security risks.

Java SE 8 is the recommended Java SDK because it provides the latest features and security updates. You can install it by installing the IBM SDK, Java Technology Edition, Version 8 package to the package group that contains WebSphere Liberty.

**Note:** The Liberty runtime environment searches for the `java` command in this order: `JAVA_HOME` property, `JRE_HOME` property, and system `PATH` property.

For more information about supported Java environments, and where to get them, see “Minimum supported Java levels” on page 1481.

4. Optional: Upgrade your Liberty installation to an unlimited production edition. Upgrading your license gives you full access to IBM support in an unlimited test or production environment. For more information, see “Upgrading Liberty installations” on page 847.

## What to do next

After you install Liberty from a ZIP file, you can further customize your environment by installing additional assets; see “Installing Liberty Repository assets” on page 852.

**IBM i** On the IBM i platform, after you extract the distribution image, you can configure all servers to run as jobs in the batch subsystem under the `QEJBSVR` user profile that is provided with the product. For more information, see “Configuring the Liberty server to start as a job in the `QWAS85` subsystem on IBM i” on page 850.

## List of installation ZIP archive files

**Distributed operating systems**

**IBM i**

**8.5.5.6**

You can install Liberty by extracting a ZIP archive file. After you install Liberty, you can install additional features by using the `installUtility` command.

The following sections list Liberty ZIP archive files. For more information about each ZIP file, see the related product asset description on the WASdev community download page.

### WAS Liberty V8.5.5.x Kernel

`wlp-kernel-<version>.zip`

This basic ZIP file includes only the kernel of the Liberty server and no features.

## WAS Liberty V8.5.5.x with Java EE 7 Web Profile

### wlp-webProfile7-<version>.zip

This ZIP file includes the Liberty server runtime environment and features that support the Java EE 7 Web Profile.

### wlp-webProfile7-java8-<platform>-<architecture>-<version>.zip

These ZIP files include the Liberty server runtime environment, IBM SDK Java Technology Edition Version 8, and features that support the Java EE 7 Web Profile. There are individual ZIP files for each available platform and architecture. You can later update the IBM Java SDK; for more information, see “Updating the Java software development kit in a Liberty ZIP archive installation” on page 846.

## WAS Liberty V8.5.5.x with Java EE 7 Full Platform

### wlp-javaee7-<version>.zip

This ZIP file includes the Liberty server runtime environment and features that support Java EE 7.

## WAS Liberty V8.5.5.x with Java EE 7 Application Client

### wlp-javaeeClient7-<version>.zip

This ZIP file includes the Liberty server runtime environment and the Java EE 7 application client.

8.5.5.9

## WAS Liberty V8.5.5.x with OSGi Applications

### wlp-osgi-<version>.zip

This ZIP file includes the Liberty server runtime environment and features that support OSGi applications.

## Features included in the installation ZIP files

The following table compares the features that are included in each of the available ZIP files. For more information about these features, see “Liberty features” on page 483.

Table 65. Features that are included in the installation ZIP files

Feature	WAS Liberty V8.5.5.x Kernel	WAS Liberty V8.5.5.x with Java EE 7 Web Profile	WAS Liberty V8.5.5.x with Java EE 7 Full Platform	WAS Liberty V8.5.5.x with Java EE 7 Application Client	WAS Liberty with Micro profile	WAS Liberty V8.5.5.x with OSGi Applications
webProfile-7.0		△				
javaee-7.0			△			
appSecurityClient-1.0 javaeeClient-7.0			△	△		

Table 65. Features that are included in the installation ZIP files (continued)

Feature	WAS Liberty V8.5.5.x Kernel	WAS Liberty V8.5.5.x with Java EE 7 Web Profile	WAS Liberty V8.5.5.x with Java EE 7 Full Platform	WAS Liberty V8.5.5.x with Java EE 7 Application Client	WAS Liberty with Micro profile	WAS Liberty V8.5.5.x with OSGi Applications
blueprint-1.0 distributedMap-1.0 el-3.0 jaxrs-2.0 jdbc-4.1 jndi-1.0 jpa-2.1 jsf-2.2 jsonp-1.0 jsp-2.3 osgiBundle-1.0 sessionDatabase-1.0 servlet-3.1 websocket-1.1						⬇
appSecurity-2.0 collectiveMember-1.0 ldapRegistry-3.0 localConnector-1.0 monitor-1.0 requestTiming-1.0 restConnector-1.0 ssl-1.0 webCache-1.0		⬇	⬇			⬇

## Applying a fix pack to a Liberty ZIP archive installation

Distributed operating systems

IBM i

8.5.5.6

Liberty offers a ZIP archive-based installation as an alternative to using IBM Installation Manager. If you installed Liberty from the ZIP archive file and want to upgrade to the latest fix pack version, you can apply a new fix pack archive to a new location, and move any required user files and server configuration data.

### About this task

If you used IBM Installation Manager to install Liberty, you must use Installation Manager to apply a fix pack.

**Important:** You must apply a new fix pack archive to a new location.

### Procedure

1. Install the new runtime environment by downloading the ZIP archive file from WASdev.net and extracting it. For more information, see “Installing Liberty by extracting a ZIP archive file” on page 842.
2. Optional: Upgrade the new installation to an edition with advanced functionality. If you upgraded your previous installation, upgrade to a product edition with at least the same level of functionality to reduce the chance of any incompatibilities. For example, if you previously upgraded to WebSphere Application Server Liberty (base), upgrade to that same edition or WebSphere Application Server Liberty Network Deployment. For more information, see “Upgrading Liberty installations” on page 847.

- Optional: Install Liberty Repository assets to customize the new environment. For more information, see “Installing Liberty Repository assets” on page 852.
- Move any user data and server configurations. Liberty defines two locations for storing user-generated content and server configurations:

- WLP\_USER\_DIR; The location of server configuration files, including shared resources.
- WLP\_OUTPUT\_DIR; The location of resources generated by the server. For example, log files and temporary disk storage.

If the WLP\_USER\_DIR environment variable has been set on your system, then the new runtime environment will continue to use the same location. This results in no backup of server configuration data. To ensure that your server configuration is backed up, copy the directory referenced by WLP\_USER\_DIR to a new location on your file system. To protect the original environment, change the value of WLP\_USER\_DIR to point to the new location. During uninstallation, reset the value of WLP\_USER\_DIR to the location of the original server configuration.

If WLP\_USER\_DIR has not been set, the server configuration and shared resources are stored in the `usr` directory at the root of the server's runtime environment (for example, `<liberty_server_runtime_root>/usr`). During uninstallation of the runtime environment, you can reset the WLP\_USER\_DIR environment variable.

If the WLP\_OUTPUT\_DIR environment variable is set on your system, the new server also uses this location. This can result in old log files being overwritten. To ensure that old log files are protected, either update or unset the WLP\_OUTPUT\_DIR environment variable. During uninstallation, reset this value to its original value.

If the WLP\_OUTPUT\_DIR value is not set, the default location is in the server root directory (for example `<liberty_server_runtime_root>/usr/servers/<serverName>`). If the new runtime environment is installed to a new location, no updates are required during installation or uninstallation because logs continue to appear under the `usr/servers/<serverName>/logs` directory of each respective installation.

**Note:** If the `server.xml` file, or any included XML configuration file, references another resource outside the server configuration directory, these resources must also be copied across, or the references will need to be updated. This also applies to any resources that the application references directly, such as references to hardcoded paths on file systems. During uninstallation of the fix pack, these values can be manually reset to their original values.

- Start the new server. Run `<liberty_VX+>/bin/server start <server_name>`.

## What to do next

After you verify that the new installation works correctly, you can remove the old installation. In the previous installation, stop all servers, then delete the `wlp` directory.

## Updating the Java software development kit in a Liberty ZIP archive installation

Distributed operating systems

IBM i

8.5.5.6

If you installed Liberty with IBM SDK, Java Technology Edition, Version 8 by extracting a ZIP archive file, you can update the Java software development kit (SDK) to incorporate Java service releases and interim fixes.

### About this task

The directories in the following procedures are based on Liberty installations that were installed by extracting a `wlp-webProfile7-java8-<platform>-<architecture>-<version>.zip` file, available from IBM Fix Central or WASdev.net. If you separately installed or moved the IBM Java SDK to a different location, you can update the SDK by replacing the files at their current location.

## Procedure

**8.5.5.7** For Version 8.5.5.7 or later:

- Update the Java SDK in a Liberty ZIP archive installation.
  1. Download the IBM SDK, Java Technology Edition, Version 8 ZIP archive file from IBM Fix Central. If you are applying a Java service release or interim fix that is associated with a certain APAR, see the APAR text for download information.
  2. Extract the Java SDK ZIP archive file to a local directory.
  3. Back up the `wlp/java/java` directory in your Liberty installation by renaming the folder. For example, rename the `wlp/java/java` directory to `wlp/java/java_backup`.
  4. Copy the `sdk` folder from the directory where you extracted the Java SDK ZIP archive file to the `wlp/java` directory. Rename the new `wlp/java/sdk` directory to `wlp/java/java`.

**8.5.5.6** For Version 8.5.5.6 only:

- Update the Java SDK in a Liberty ZIP archive installation.
  1. Download the IBM SDK, Java Technology Edition, Version 8 ZIP archive file from IBM Fix Central. If you are applying a Java service release or interim fix that is associated with a certain APAR, see the APAR text for download information.
  2. Extract the Java SDK ZIP archive file to a local directory.
  3. Back up the `wlp/java` directory in your Liberty installation by renaming the folder. For example, rename the `wlp/java` directory to `wlp/java_backup`.
  4. Copy the `sdk` folder from the directory where you extracted the Java SDK ZIP archive file to the `wlp` directory. Rename the new `wlp/sdk` directory to `wlp/java`.
  5. Copy the `java.env` file from the `wlp/java_backup` backup directory to the `wlp/java` directory.

## What to do next

After you verify that IBM SDK, Java Technology Edition, Version 8 updated successfully, delete the Java backup directory.

## Upgrading Liberty installations

Distributed operating systems

IBM i

8.5.5.5

You can upgrade the edition or license of Liberty by using a self-extracting Java archive (JAR) file.

## Before you begin

To download the upgrade JAR files from IBM Fix Central or Passport Advantage, you must be enrolled as a customer.

Your system must meet the operating system and Java requirements for using Liberty. See System Requirements for WebSphere Application Server V8.5.5.

**IBM i**

For the IBM i platform, the minimum supported Java level is installed at one of the following locations:

- `/QOpenSys/QIBM/ProdData/JavaVM/jdk626/32bit`
- `/QOpenSys/QIBM/ProdData/JavaVM/jdk626/64bit`

## About this task

Liberty offers installation from a ZIP archive file or a self-extracting Java archive (JAR) file as an alternative to using IBM Installation Manager. If you installed Liberty using one of these archive files, you can upgrade the product license by applying an executable JAR file, available from IBM Fix Central or the Passport Advantage website.

You might upgrade Liberty in the following example scenarios:

- Upgrade a trial edition or other unsupported edition to a supported production edition.
- Upgrade a basic edition to a more advanced edition for access to advanced features.

You can upgrade WebSphere Application Server Liberty offerings in the following paths:

*Table 66. Liberty offering upgrade paths*

License JAR file	Source edition	Target edition
wlp-core-license.jar	Unsupported ZIP or JAR file installations WebSphere Application Server Liberty Core Trial	WebSphere Application Server Liberty Core
wlp-base-license.jar	Unsupported ZIP or JAR file installations WebSphere Application Server Liberty Trial WebSphere Application Server Liberty Core WebSphere Application Server Liberty for Developers WebSphere Application Server Liberty - Express	WebSphere Application Server Liberty
wlp-nd-license.jar	Unsupported ZIP or JAR file installations WebSphere Application Server Liberty Network Deployment Trial WebSphere Application Server Liberty Core WebSphere Application Server Liberty for Developers WebSphere Application Server Liberty - Express WebSphere Application Server Liberty	WebSphere Application Server Liberty Network Deployment

For example, an installation of WebSphere Application Server Liberty Core also requires features from WebSphere Application Server Network Deployment. If you have the required entitlement, you can download the wlp-nd-license.jar license file and apply it to your Liberty installation, which upgrades the license from WebSphere Application Server Liberty Core to WebSphere Application Server Network Deployment. Upgrading the license does not install any features from WebSphere Application Server Network Deployment, but it enables you to install WebSphere Application Server Network Deployment features to your Liberty installation if needed.



## Procedure

1. Download the license upgrade archive for the edition that you want to upgrade to from IBM Fix Central or the IBM Passport Advantage website. You can apply the license JAR files to all fix packs of Liberty ZIP or JAR file installations.

For a list of the JAR files on IBM Fix Central, see license JAR files on the Fix Central website.

For a list of Passport Advantage part numbers, see How to download WebSphere Application Server Liberty Core V8.5.5 from Passport Advantage Online.

2. Apply the license archive to an existing Liberty installation. For example, to apply the license by using the installation wizard and the `wlp-nd-license.jar` file, run the following command:

```
java -jar wlp-nd-license.jar
```

## Applying an interim fix to a Liberty archive installation

Distributed operating systems

IBM i

Liberty offers a self-extracting archive-based installation as an alternative to using IBM Installation Manager. If you installed Liberty using the self-extracting archive, and want to install an interim fix, you can apply an executable JAR file.

### About this task

If you used IBM Installation Manager to install Liberty, you must use Installation Manager to apply an interim fix.

An archive-based interim fix is named `<Liberty profile level>-WS-WASProd_WLPArchive-<fix type><fix id>.jar`

- `<Liberty profile level>` refers to a 4-digit fix pack level identifier, which indicates the minimum level to which the fix applies. For example, 8.5.5.0.
- `<fix type>` refers to the type of fix. For example, IF is used for an interim fix, and TF for a diagnostic fix
- `<fix id>` refers to the APAR reference number. If a diagnostic fix is provided before an APAR is opened, the `<fix id>` is based on the PMR reference number.

Each interim fix is installed by executing the relevant JAR file, which extracts the content into the Liberty base folder (`/wlp`).

**Note:** When the interim fix is applied, no backup data is created. If you want to back out an interim fix, you must manually remove or restore files from the `/wlp` folder.

Each interim fix is provided with a `readme.txt` file, which contains backup and restore instructions specific to the fix content, in a section titled **Directions to apply fix**. If the `readme.txt` file does not specify any requirement to back up files, you can extract the fix and then restart the server at any time with the `--clean` parameter as a launch option.

## Procedure

1. Optional: If the fix contains files that will overwrite existing files, stop all servers that are running on the system.
2. Optional: If the `readme.txt` file indicates that a backup is required, create a backup of the `lib/features/component.mf` files. File locations are relative to your Liberty profile installation root.
3. Open a console and direct it to the location of your interim fix JAR file.
4. To view available options, run `java -jar interim_fix_jar_file -help`, where `interim_fix_jar_file` is the name of the executable JAR file that contains the interim fix. The following launch options are available for the JAR file:

### **--installLocation [LibertyRootDir]**

The absolute or relative location of the Liberty installation directory.

By default the JAR file looks for a `wlp` directory in its current location. If your Liberty profile installation location is not the `wlp` folder, or is not in the same directory as the JAR file, then you can use this option to change where the JAR file will patch. `[LibertyRootDir]` can either be relative to the location of the JAR file, or an absolute file path.

### **--suppressInfo**

The only messages output from the JAR file will be error messages or confirmation that patching is complete.

5. Run the JAR file, using the `--installLocation` option to specify the installation location. For example, `java -jar interim_fix_jar_file --installLocation`. Optionally include `--suppressInfo`.
6. Start all Liberty profile servers with the `--clean` parameter as a launch option. For example, `server start --clean`. You use the `--clean` option only once; all subsequent server starts do not require it.

## **Removing an interim fix from a Liberty archive install**

Distributed operating systems

IBM i

If you installed Liberty by using the self-extracting archive, then to remove an interim fix you must manually remove files, and restore files, from the `/wlp` folder.

### **About this task**

If you used IBM Installation Manager to apply an interim fix, you can use Installation Manager to remove the interim fix.

The current set of fixes installed on a Liberty profile can be found in the `/lib/fixes` directory.

Each interim fix is provided with a `readme.txt` file, which contains backup and restore instructions specific to the fix content, in a section titled **Directions to apply fix**. If the `readme.txt` file does not include any requirement to back up files, you can extract the fix and then restart the server at any time with the `--clean` parameter as a launch option.

### **Procedure**

1. Stop all servers running on the system. For more information, see [Starting and stopping a server](#).
2. Delete or replace the files as detailed in the `readme.txt` file. File locations are relative to your Liberty profile install root. For example:
  - `lib/com.ibm.ws.component_1.0.0.20120803-1356.jar`
  - `lib/fixes/8.5.0.0-WASProd_WLPArchive-IFPM1111_8.5.0.20120803-1356.xml`
3. Start all Liberty profile servers with the `--clean` parameter as a launch option. For example, `server start --clean`. You only need to use the `--clean` option once. All subsequent server starts will not require it.

### **What to do next**

You can reapply the fix by following the instructions in “Applying an interim fix to a Liberty archive installation” on page 849.

## **Configuring the Liberty server to start as a job in the QWAS85 subsystem on IBM i**

IBM i

You can optionally use the iAdmin POSTINSTALL command to configure the Liberty server to start as a job in the QWAS85 subsystem and to run under the QEJBSVR user profile.

## About this task

The iAdmin POSTINSTALL command configures the server start command to start servers as jobs in the QWAS85 subsystem. Additionally, the task can be used to:

- Configure Liberty to run servers under the QEJBSVR user profile.
- Configure the default JDK location by setting WLP\_DEFAULT\_JAVA\_HOME in file wlp/etc/default.env to the location of the 32 bit version of the minimum supported Java level.
- Add an entry for the product in the IBM i native product registry.
- Create IBM i native libraries and objects such as the QWAS85 subsystem and the QEJBSVR user profile.

Call the iAdmin POSTINSTALL command only after Installing Liberty by extracting an archive file or when Installing Liberty resources by using the job manager. When installed using the IBM Installation Manager, the Liberty server is already configured to start as a job in the QWAS85 subsystem and to run under the QEJBSVR user profile.

**Note:** You must have \*ALLOBJ and \*SECADM special authority to use the iAdmin POSTINSTALL command.

## Procedure

1. On the IBM i command line, run the STRQSH command to start the Qshell.
2. In the resulting Qshell session, change the directory to the wlp directory.
3. Run the following command.

```
lib/native/os400/bin/iAdmin POSTINSTALL
```

## What to do next

Install the WebSphere Application Server group PTF to apply required PTFs for other products on which WebSphere Application Server relies. See [IBM i Installing the WebSphere Application Server group PTF on IBM i](#).

## Uninstalling Liberty application-serving environment from IBM i operating systems

IBM i

By calling the iAdmin PREUNINSTALL command, you can remove any IBM i native objects and libraries created by the iAdmin POSTINSTALL command. Then you can manually remove the files comprising your Liberty profile application-serving environment.

## Before you begin

After installing the Liberty application-serving environment by executing a JAR file on IBM i, you might have used the iAdmin POSTINSTALL command to configure the Liberty server to launch as a job in the QWAS85 subsystem and to run under the QEJBSVR user profile. If so, you must remove the IBM i native objects and libraries created at that time before you manually remove the files that comprise your Liberty profile application-serving environment.

**Note:** You must have \*ALLOBJ and \*SECADM special authority to use the iAdmin POSTINSTALL command.

## Procedure

1. On the IBM i command line, run the STRQSH command to start the Qshell.
2. In the resulting Qshell session, change the directory to the wlp directory.
3. Run the following command.

```
lib/native/os400/bin/iAdmin PREUNINSTALL
```

## What to do next

Manually remove the files that comprise your Liberty profile application-serving environment.

---


## Installing Liberty Repository assets



8.5.5.2

You can use a variety of tools to install the assets that are in the Liberty Repository or download them directly from the WASdev website.

### Before you begin

**Important:** Product documentation that is marked with the  icon indicates information about assets that are available only from the Liberty Repository.

The Liberty Repository provides an online mechanism to deliver Liberty and additional content, enabling a single point of access for various asset types. For more information, see “Liberty Repository” on page 573.

#### Note:

- To install assets from the Liberty Repository, you must have Liberty for WebSphere Application Server Version 8.5.5.2 or later.
- To access the IBM WebSphere Liberty Repository with limited internet access or through a firewall, ensure that you have access to the following hosts and ports:
  - public.dhe.ibm.com on port 443
  - asset-websphere.ibm.com on port 443

### About this task

You can use several methods to access and install Liberty Repository assets. Some methods, such as Installation Manager, require the completion of specific steps in the installation sequence before you can install the applicable assets.

## Procedure

1. Review the list of available assets in the Liberty Repository. See the Downloads page on WASdev.net. You can sort assets by date, name, or rating, or you can filter assets by asset type or Liberty edition. For a list of Liberty features, see “Liberty features” on page 483
2. Review the installation methods for the Liberty Repository assets. Use the following table to determine the appropriate installation methods for each asset type:

Table 67. Installation methods for Liberty Repository asset types

Asset type	installUtility command	featureManager command	configUtility command	Installation Manager	Developer tools <sup>a</sup>	WASdev site <sup>b</sup>
Addons	✓	✓		✓	✓	✓

Table 67. Installation methods for Liberty Repository asset types (continued)

Asset type	installUtility command	featureManager command	configUtility command	Installation Manager	Developer tools <sup>a</sup>	WASdev site <sup>b</sup>
Admin scripts					✓	
Config snippets			✓		✓	✓
Features	✓	✓		✓	✓	
Open source integrations	✓			✓	✓	✓
Products				✓ <sup>c</sup>	✓	✓
Product samples	✓			✓	✓	✓
Tools						✓

**Note:**

- Developer tools are available only for the Linux, Mac OS, and Windows platforms.
- You can download assets from the WASdev.net website to install separately.
- You can install product runtimes from Installation Manager, but they are not in the Liberty Repository.

- Choose the installation method for the Liberty Repository assets that fits your usage scenario.
  - 8.5.5.6** **installUtility** command: Finds, obtains information about, and installs a variety of assets that are in a local directory-based repository, an instance of the Liberty Asset Repository Service, or the IBM WebSphere Liberty Repository. You can also download Liberty Repository assets to your local system for offline use. The **installUtility** command downloads and installs all dependencies for the asset. For more information, see “Installing assets using the installUtility command.”
  - featureManager** command: Installs Liberty Repository features and add-ons to an existing installation of Liberty or downloads them to your local system for offline use. For more information, see “Installing assets using the featureManager command” on page 864. For Version 8.5.5.6 and later, use the **installUtility** command instead because it enables you to work with more asset types and from multiple repositories.
  - 8.5.5.5** **configUtility** command: Downloads config snippet assets and enables you to replace configuration snippet variables with your input values. For more information, see “Downloading and customizing configuration snippets from the command line” on page 958.
  - IBM Installation Manager: Installs Liberty Repository assets during the installation or an upgrade to a new version of Liberty. You can use the Installation Manager graphical user interface, command-line interface, or the Installation Manager in silent mode with a response file. For more information, see “Installing assets using Installation Manager” on page 872.
  - Distributed operating systems** Developer tools: Developer tools for WebSphere Application Server, such as WebSphere Developer Tools, provide a graphical user interface that you can use to install Liberty Repository assets. For more information, see “Installing assets by using developer tools” on page 873.
  - WASdev site: You can download a subset of the Liberty Repository assets directly from the WASdev site. See the installation instructions for each downloadable asset on WASdev.net.

## Installing assets using the installUtility command



8.5.5.6

You can use the **installUtility** command to install assets in your Liberty environment and view required asset information.

## About this task

After you install Liberty, you can install assets by running the **installUtility** command. The **installUtility** command automatically installs asset dependencies.

You can install assets from the following repositories:

- The IBM WebSphere Liberty Repository, a public IBM-hosted repository that is accessible through the internet. For more information, see “Liberty Repository” on page 573.
- The Liberty Asset Repository Service, an open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. For more information, see the WASdev/tool.lars project on GitHub.
- Local directory-based repositories that are created by using the **installUtility download** action. For more information, see “Downloading assets using the installUtility command” on page 860.

By default, the **installUtility** command is configured to install assets only from the Liberty Repository. If you want to install assets from a local directory-based repository or an instance of the Liberty Asset Repository Service, you must configure these repositories in the `repositories.properties` file. For more information, see “Configuring repositories and proxy settings for the installUtility command” on page 862.

For a list of the actions and options for the **installUtility** command, see “installUtility command” on page 855.

## Procedure

1. Review the available assets and obtain the *shortName* for each asset that you want to install. You can use the **installUtility** command to search for and review assets, or you can find them on the Downloads page on WASdev.net.
  - To review assets and obtain the asset *shortName* by using the **installUtility** command, use the **installUtility find** command.
    - To find assets that are applicable to your configuration, use the *searchString* option. The following example searches for all assets that match the search string `jca`:

```
installUtility find jca
```
    - To find a specific type of asset, use the **--type=[feature|sample|opensource|addon|all\*]** option. The following example searches for features that match the search string `jca`:

```
installUtility find jca --type=feature
```
  - To review assets and obtain the asset *shortName* from the WASdev website, see the asset details page on the WASdev website for each feature that you want to install. The *shortName* is found in the installation instructions of the asset details page; for example: the Portlet Container *shortName* is `portlet-2.0`.

For a list of all Liberty features, see “Liberty features” on page 483.

2. Run the **installUtility install** command to install the assets.

For example, the following command installs the `jca-1.6` feature:

```
installUtility install jca-1.6
```

To install multiple assets, separate each *shortName* with a space. For example, the following command installs both the `jca-1.6` feature and the `adminCenter-1.0` feature:

```
installUtility install jca-1.6 adminCenter-1.0
```

**Tip:** To install all features that apply your edition, you can install a feature bundle `addon`.

For example, to install all features for WebSphere Application Server Network Deployment Liberty, install the `ndControllerBundle` `addon` on servers that manage Liberty collectives and the `ndMemberBundle` `addon` on servers that are clustered and auto-scaled in a Liberty collective.

```
installUtility install ndControllerBundle
```

For example, to install all features for WebSphere Application Server Liberty Core, install the `libertyCoreBundle` addon.

```
installUtility install libertyCoreBundle
```

For a list of available addons, see Addon downloads on WASdev.

## installUtility command

8.5.5.6

Use the **installUtility** command to find, obtain information about, and install assets that are in a directory-based repository, an instance of the Liberty Asset Repository Service, or the IBM WebSphere Liberty Repository.

### Asset sources

The **installUtility** command can access assets in the following repositories:

- The IBM WebSphere Liberty Repository, a public IBM-hosted repository that is accessible through the internet. For more information, see “Liberty Repository” on page 573.
- The Liberty Asset Repository Service, an open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. For more information, see the WASdev/tool.lars project on GitHub.
- Local directory-based repositories that are created by using the **installUtility download** action. For more information, see “Downloading assets using the installUtility command” on page 860.

8.5.5.8

You can use the **installUtility** command to work with directory-based repository assets directly from compressed repositories, such as `wlp-featureRepo-<version>.zip`, without extracting the archives. For more information about the `wlp-featureRepo-<version>.zip` file, see “Downloading assets using the installUtility command” on page 860.

### Syntax

The command syntax is as follows:

```
installUtility action [options]
```

The *action* variable can take one of the following values:

#### download

Download assets from the repositories.

**find** Find assets in the repositories that are applicable to your configuration, or view detailed information about assets.

**install** Install assets or an enterprise subsystem archive (ESA) file to the run time, or deploy a server package and install the required features of the package.

#### testConnection

Test the repository connection.

#### uninstall

Uninstall features by specifying either the feature short name or the feature symbolic name. You can specify multiple features, separated by a space.

**Note:** Ensure that all server processes are stopped before you uninstall a feature.

#### viewSettings

View a template for configuring repositories or a proxy, or view the settings for the configured repositories or proxy.

**8.5.5.7** For Version 8.5.5.7 and later, this action also validates the settings for any configured repositories or proxy in the `repositories.properties` file.

**help** Display help information for a specified action.

## Options

The following options are available for the **installUtility download** command:

**--acceptLicense**

Accepts the license agreement.

**--viewLicenseAgreement**

View the license agreement.

**--viewLicenseInfo**

View the license agreement.

**--location=*directoryPath***

Specify the destination directory for the downloaded assets. This option is required.

**--overwrite**

Use this option to overwrite the existing files when you download assets to the local directory. The default behavior without the option is to ignore all the existing files.

**--verbose**

Use this option to display additional information during the download.

**name** Specify asset IDs to download one or more of the following assets, separating multiple asset IDs with a space:

- Features
- Addons
- Open source integrations
- Samples

The following options are available for the **installUtility find** command:

**8.5.5.7** **--from**

Specify a single directory-based repository as the source of the assets for the **installUtility** command.

**8.5.5.8** The directory-based repository can be an uncompressed folder or a compressed archive file, such as `wlp-featureRepo-<version>.zip`.

To search for assets in multiple directory-based repositories, you must configure the repositories in the `repositories.properties` file. For more information, see “Configuring repositories and proxy settings for the `installUtility` command” on page 862.

**--showDescriptions**

Displays the description for each of the features that are found by the search.

**--type=[feature | sample | opensource | addon | all\*]**

Searches for the specified type of assets.

**--name**

Searches the asset name for the specified *searchString*.

**--verbose**

Use this option to display any available additional information while the action runs.

**searchString**

Finds assets that are applicable to your configuration. If you do not specify a search string, the command searches for all applicable assets.



The following options are available for the **installUtility install** command:

**--to=install\_option**

The `install_option` option can take one of the following values:

- `usr`: The feature is installed as a user feature. This value is the default value.
- `extension`: The location to install the feature. You can install the feature to any configured product extension location.

8.5.5.7

**--from**

Specify a single directory-based repository as the source of the assets for the **installUtility** command.

8.5.5.8

The directory-based repository can be an uncompressed folder or a compressed archive file, such as `wlp-featureRepo-<version>.zip`.

To install assets from multiple directory-based repositories, you must configure the repositories in the `repositories.properties` file. For more information, see “Configuring repositories and proxy settings for the `installUtility` command” on page 862.

**--acceptLicense**

Indicate acceptance of license terms and conditions.

**--viewLicenseAgreement**

View the license agreement.

**--viewLicenseInfo**

View the license information.

**--verbose**

Use this option to display any available additional information while the action runs.

**name** Specify one or more assets that you want to install. You can specify the assets in the following ways:

**Asset IDs**

You can specify asset IDs to install one or more of the following assets, separating multiple asset IDs with a space:

- Features
- Addons
- Open source integrations
- Samples

**Server name**

You can install features based on an existing server in the same Liberty environment by specifying the server name. The command installs any applicable features that are defined in the `server.xml` file of that server that are not already installed in the Liberty environment.

**Server package**

You can specify a server package that you created by using the `server package --include=usr` command. The server package deploys, and the required features, which are defined in the `server.xml` file, are installed from the repositories.

8.5.5.7

**server.xml file**

To install features based on a server configuration file, you can specify a path to any local `server.xml` file, such as `C:\localDir\server.xml`. The command installs any applicable features that are defined in the `server.xml` file that are not already installed in the Liberty environment.

**.esa file**

Install a feature from an `.esa` file on your local file system. This action uses the

OSGI-INF/SUBSYSTEM.MF manifest file from the .esa file as a new feature manifest. The manifest file specifies the name, contents, and dependencies of a feature. If you specify the **--to** option, **installUtility** copies the manifest file into the `${wlp.user.dir}/extensions/lib/features` directory, or the product extension directory. All of the bundles for the subsystem are extracted into the `${wlp.user.dir}/extensions/lib` directory and renamed as `{bundle symbolic name}_{bundle version}.jar`. License files, checksum files, globalization files, and other subsystem content are extracted to the location defined in the subsystem manifest.

The following option is available for the **installUtility testConnection** command:

**repoName**

Specifies the name of the repository to be tested. If not specified, all repositories that are specified in the `repositories.properties` file are tested.

The following option is available for the **installUtility uninstall** command:

**8.5.5.8 --force**

Uninstall the specified feature regardless of whether other installed features have dependencies on it.

**Note:** Uninstalling a feature that is required by other installed features might cause those features to stop working and might prevent servers from running correctly.

**--noPrompts**

Uninstalls the feature without prompts. The default is false.

**--verbose**

Use this option to display additional information during the uninstallation.

**name** Specify one or more features to uninstall, separating multiple names with a space. You can specify the following options:

- The short name of the subsystem archive (ESA file), such as `adminCenter-1.0`.
- The symbolic name of the subsystem archive (ESA file), such as `com.ibm.websphere.appserver.adminCenter-1.0`.

**8.5.5.8** You can uninstall user features and product extensions by prefixing the feature name with the extension followed by a colon, such as `usr:webCacheMonitor-1.0`.

**8.5.5.7** The following option is available for the **installUtility viewSettings** command:

**--viewValidationMessages**

Use this option to display the detailed messages from the validation of the configured `repositories.properties` file. Each message contains an error code, the line number where the error was found, and the cause of the error.

## Usage examples

Use the following example to display help information for the **install** action:

```
installUtility help install
```

Use the following example to install a user feature:

```
installUtility install my_feature --to=usr
```

Use the following example to install a user feature to the *my\_extension* product extension location:

```
installUtility install my_feature --to=my_extension
```

Use the following example to install multiple features:

```
installUtility install feature1 feature2 feature3
```

Use the following example to install a sample:

```
installUtility install mongoDBSample
```

**8.5.5.7** Use the following example to install a feature bundle addon from a single local directory-based repository:

```
installUtility install ndMemberBundle|libertyCoreBundle --from=c:\download\wlp-featureRepo-8.5.5.7
```

**8.5.5.8** Use the following example to install a feature bundle addon from a single compressed directory-based repository archive:

```
installUtility install ndMemberBundle|libertyCoreBundle --from=c:\download\wlp-featureRepo-8.5.5.8.zip
```

Use the following example to install a server:

```
installUtility install myServer
```

Use the following example to install a server package compressed .jar file:

```
installUtility install c:\temp\myServer.jar
```

Use the following example to install a local .esa file:

```
installUtility install c:\temp\myFeature.esa
```

Use the following example to install a server package compressed file:

```
installUtility install c:\temp\myServer.zip
```

Use the following example to find assets that are applicable to your configuration:

```
installUtility find searchString
```

Use the following example to find samples that are applicable to your configuration:

```
installUtility find searchString --type=sample
```

Use the following example to find a particular feature that is applicable to your configuration:

```
installUtility find webCacheMonitor-1.0 --name --type=feature
```

Use the following example to list detailed information:

```
installUtility find searchString --showDescriptions
```

Use the following example to download required dependencies from the repositories to a local directory:

```
installUtility download feature_shortName --location=c:\temp\download --acceptLicense
```

Use the following example to test the connection to the Liberty Repository:

```
installUtility testConnection default
```

Use the following example to uninstall a feature:

```
installUtility uninstall adminCenter-1.0
```

**8.5.5.8** Use the following example to uninstall a user feature or product extension:

```
installUtility uninstall usr:webCacheMonitor-1.0
```

Use the following example to uninstall multiple features:


```
installUtility uninstall feature1 feature2 feature3
```

Use the following example to uninstall multiple features without prompts:

```
installUtility uninstall --noPrompts feature1 feature2 feature3
```

## Return codes

Table 68. Return codes and explanations.

Return code	Explanation
0	The command successfully completed the requested operation.
20	One or more arguments are not valid.
21	A runtime exception occurred because of one or more of the following conditions: <ul style="list-style-type: none"><li>• A runtime exception occurred during the installation of the .esa subsystem archive file.</li><li>• A license is not accepted or acknowledged.</li><li>• The .esa subsystem archive file did not extract correctly.</li></ul>
22	The feature to be installed exists.
23	The feature definition was not valid for one or more of the following reasons: <ul style="list-style-type: none"><li>• The feature does not have a valid manifest file.</li><li>• The version of the feature is not supported in this Liberty environment.</li><li>• The .zip or .jar file that contains the feature files does not exist.</li></ul>
24	The .esa subsystem archive file is missing content.
25	A file to be installed exists.
26	The product is not a core product, and the product extension files cannot be found.
27	The product is not a core product, and the product extension is not defined in the <code>#{wlp.install.dir}/etc/extensions/extension_name.properties</code> file.
28	The manifest files for the feature in the product extension cannot be found.
29	The feature is not valid for the current product.
 30	The <code>repositories.properties</code> file failed the validation.
33	The connection to the repository failed.
34	The repository name is not found. The provided repository name does not exist in the configuration file.
35	The action was canceled by the user.

## Downloading assets using the `installUtility` command



8.5.5.6

You can use the `installUtility` command to download assets to your local file system.

### Before you begin

Before you can access the IBM WebSphere Liberty Repository using the `installUtility` command, you must install Liberty for WebSphere Application Server.

## About this task

After you install Liberty, you can download assets to your local file system by running the **installUtility** command. Assets can be downloaded to a new directory, which you can then use as a repository, or to an existing directory-based repository.

By default, the **installUtility** command is configured to download assets only from the public IBM WebSphere Liberty Repository. If you want to download assets from a local directory-based repository or an instance of the Liberty Asset Repository Service, you must configure these repositories in the `repositories.properties` file. For more information, see “Configuring repositories and proxy settings for the `installUtility` command” on page 862.

For a list of the actions and options for the **installUtility** command, see “`installUtility` command” on page 855.

**Tip:** Rather than downloading individual assets, you can download a `wlp-featureRepo-<version>.zip` file from IBM Fix Central. The `.zip` file contains a directory-based repository of all features and add-ons for the particular fix pack of Liberty that you download from IBM Fix Central. Extract the archive file to populate a local directory-based repository or an instance of the Liberty Asset Repository Service with the feature and add-on ESA files. Assets that are not included in the `.zip` file must be downloaded by using the **installUtility** command.

**8.5.5.8** For Version 8.5.5.8 and later, you can use the **installUtility** command to work with repository assets directly from the `wlp-featureRepo-<version>.zip` file and other compressed directory-based repositories without extracting the archives.

## Procedure

1. Review the available assets and obtain the *shortName* for each asset that you want to download. You can use the **installUtility** command to search for and review assets, or you can find them on the Downloads page on WASdev.net.
  - To review assets and obtain the asset *shortName* using the **installUtility** command, use the **installUtility find** command.
    - To find assets that are applicable to your configuration, use the *searchString* option. The following example searches for all assets that match the search string `adminCenter`:

```
installUtility find adminCenter
```
    - To find a specific type of asset, use the **--type=[feature|sample|opensource|addon|all\*]** option. The following example searches for features that match the search string `adminCenter`:

```
installUtility find adminCenter --type=feature
```
  - To review assets and obtain the asset *shortName* from the WASdev website, see the asset details page on the WASdev website for each feature that you want to install. The *shortName* is found in the installation instructions of the asset details page; for example: the Portlet Container **shortName** is `portlet-2.0`.
2. Run the **installUtility download** command to download the assets. You must specify a target directory for the downloaded assets on the **--location** option.

For example, the following command downloads the `adminCenter-1.0` feature to the `c:\temp\download` directory:

```
installUtility download adminCenter-1.0 --location=c:\temp\download
```

You can download multiple assets at once by separating each asset *shortName* with a space. For example, the following command downloads both the `jca-1.6` feature and the `adminCenter-1.0` to the `c:\temp\download` directory:

```
installUtility download jca-1.6 adminCenter-1.0 --location=c:\temp\download
```

## What to do next

After you download assets to your local file system, you can add a local directory to your repository configuration so that you can install assets from the directory. For more information about configuring repositories, see “Configuring repositories and proxy settings for the `installUtility` command.”

## Configuring repositories and proxy settings for the `installUtility` command

8.5.5.6

In the `repositories.properties` file, you can configure how the `installUtility` command accesses local directory-based repositories, instances of the Liberty Asset Repository Service, and the IBM WebSphere Liberty Repository.

### About this task

The `installUtility` command can access the following repositories:

- The IBM WebSphere Liberty Repository, a public IBM-hosted repository that is accessible through the internet. For more information, see “Liberty Repository” on page 573.
- The Liberty Asset Repository Service, an open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. For more information, see the WASdev/tool.lars project on GitHub.
- Local directory-based repositories that are created by using the `installUtility download` action. For more information, see “Downloading assets using the `installUtility` command” on page 860.

Both the `installUtility` command and the `featureManager` command use the same configuration properties file, `repositories.properties`. The proxy settings are shared, but the repository settings are used only by the `installUtility` command. To manage assets and complete repository-based installation, find, or uninstallation operations, use the `installUtility` command instead of the `featureManager` command.

**Tip:** You can download or install assets from a single local directory-based repository by specifying the repository on the `--from` option. No additional configuration in the `repositories.properties` file is required. For more information, see “`installUtility` command” on page 855.

To access the IBM WebSphere Liberty Repository with limited internet access or through a firewall, ensure that you have access to the following hosts and ports:

- `public.dhe.ibm.com` on port 443
- `asset-websphere.ibm.com` on port 443

**Note:** The `installUtility` command supports only proxy servers with HTTP/HTTPS protocols.

### Procedure

1. Optional: When you first configure the repository or proxy settings, you can run the `installUtility viewSettings` command to output a configuration template for the `repositories.properties` file.  
Copy the template into a new properties file at `${wlp.install.dir}/etc/repositories.properties`. The properties file must be in ASCII format for all platforms.
2. Define the settings in the `${wlp.install.dir}/etc/repositories.properties` file.  
If you copied the template from the `installUtility viewSettings` command, modify the template by changing the example repository and proxy settings to refer to your environment. Lines that begin with a number sign (#) are not processed.
  - To disable access to the Liberty Repository, set the `useDefaultRepository` property to `false`. The Liberty Repository is enabled by default and is the last repository that is accessed when you install or download assets.

```
useDefaultRepository=false
```

- To define a repository, add a property as `repoName.url=url`. Each repository name must be unique. The defined repositories are accessed in the order that they are specified in the `repositories.properties` file. For Version 8.5.5.7 and later, you can also specify `repoName.url=file_path`.

Windows

```
dev-rep.url=http://dev.repo.ibm.com:9080/ma/v1
local-rep2.url=file:///c:/IBM/localrepo2
```

8.5.5.7

```
local-rep3.url=C:\IBM\localrepo3
```

8.5.5.8

```
local-rep4.url=C:\IBM\localrepo4.zip
```

HP-UX

Solaris

Linux

UNIX

```
dev-rep.url=http://dev.repo.ibm.com:9080/ma/v1
local-rep2.url=file:///usr/IBM/localrepo2
```

8.5.5.7

```
local-rep3.url=/usr/IBM/localrepo3
```

8.5.5.8

```
local-rep4.url=/usr/IBM/localrepo4.zip
```

- If a repository requires a user name and password, set the properties `repoName.user=userId` and `repoName.userPassword=password`.

If a user name and password are required and they are not set, you receive a prompt to provide them. For enhanced security, encode the password by using the **securityUtility encode** action; for more information, see Liberty `securityUtility` command. For Version 8.5.5.6, the password must be encoded to connect to the repository.

```
dev-rep.user=myname
dev-rep.userPassword={aes}AH5NLyd7DfGb12pK17Pw+
```

- If your system requires access to the IBM WebSphere Liberty Repository or an instance of the Liberty Asset Repository Service through a proxy server, set the `proxyHost`, `proxyPort`, `proxyUser`, and `proxyPassword` proxy properties.

For enhanced security, encode the value of the `proxyPassword` property by using the **securityUtility encode** action. If you do not set the user name and password, you receive a prompt to provide them. For Version 8.5.5.6, the password must be encoded to connect to the proxy.

For example:

```
proxyHost=my.proxy.server.ibm.com
proxyPort=9080
proxyUser=myname
proxyPassword={aes}AH5NLyd7DfGb12pK17Pw+
```

3. Save the changes to the `repositories.properties` file.
4. Review the repository and proxy settings by running the **installUtility viewSettings** command.

8.5.5.7 When you run the **installUtility viewSettings** command, the repository and proxy configuration is automatically validated. To view detailed validation messages, run the command with the **--viewValidationMessages** option. 8.5.5.7

```
>installUtility viewSettings
```

```
installUtility Settings
```

```

Properties File: c:\wlp\etc\repositories.properties
Default Assets Repository: IBM WebSphere Liberty Repository
Use Default Repository: True
```

```
Properties File Validation
```

```

Validation Results: The properties file successfully passed the
validation.
```

```
Configured Repositories
```

```

Name: dev-rep
```

```
Location: http://dev.repo.ibm.com:9080/ma/v1
User Name: myname
Password: <Unspecified>
```

#### Proxy Settings

```

Proxy Server: my.proxy.server.ibm.com
Port: 9080
User Name: myname
Password: *****
```

5. Test the repository connection by running the **installUtility testConnection** command. If you do not set the user name and password, you receive a prompt to provide them. Use the following example to test the repository connection:

```
>installUtility testConnection
Testing the connection to all configured repositories...
This process might take several minutes to complete.
```

#### Configured Repositories

```

Name: dev-rep
Location: http://dev.repo.ibm.com:9080/ma/v1
Status: Successfully connected to the configured repository.
```

You can test all configured repositories in the `repositories.properties` file at once by running the **installUtility testConnection** command with no `repoName` specified. The repository name for the Liberty Repository is `default`. To test the connection to the Liberty Repository, run the following command:

```
installUtility testConnection default
```

## What to do next

You can use the **installUtility** command to search for assets and install or download them from the configured repositories.

## Installing assets using the `featureManager` command



You can use the **featureManager** command to install Liberty Repository features in your Liberty environment.

### Before you begin

Before you can access the Liberty Repository using the **featureManager** command, you must install Liberty for WebSphere Application Server Version 8.5.5.2 or later.


### About this task

After you install Liberty, you can install Liberty Repository features by running the **featureManager** command. The **featureManager** command automatically installs asset dependencies.

**Tip:** Use the **installUtility** command instead of the **featureManager** command. With the **installUtility** command, you can manage more asset types and install, find, or download assets from multiple repositories. For more information, see “Installing assets using the `installUtility` command” on page 853.



## Procedure

-  **8.5.5.4** Review the assets that are located in the Liberty Repository and obtain the *feature\_shortName* for each asset that you want to install. The asset *feature\_shortName* is required to download and install assets. You can use the **featureManager** command to search for and review assets, or you can find them on the Downloads page on WASdev.net.
  - To review assets and obtain the asset *feature\_shortName* using the **featureManager** command, use the **featureManager find** command.
    - To find assets that are applicable to your configuration, specify a string to search for:

```
featureManager find searchString
```
    - To view detailed information, use the **--viewInfo** option; for example:

```
featureManager find searchString --viewInfo
```
  - To review assets and obtain the asset *feature\_shortName* from the WASdev website, see the asset details page on the WASdev website for each feature that you want to install. The *feature\_shortName* is found in the installation instructions of the asset details page; for example: the Portlet Container *feature\_shortName* is *portlet-2.0*.


For a list of all Liberty features, see “Liberty features” on page 483.

- Obtain the *feature\_shortName* from the asset details page on the WASdev website for each feature that you want to install. The *feature\_shortName* is found in the installation instructions of the asset details page; for example: the Portlet Container *feature\_shortName* is *portlet-2.0*.
- Run the **featureManager** command to install the assets. Run the following command:

```
bin/featureManager install feature_shortName --when-file-exists=ignore
```


To install multiple features, use spaces or commas to separate each *feature\_shortName*; for example:

```
bin/featureManager install feature_shortName1 feature_shortName2 --when-file-exists=ignore
```


-  **8.5.5.4** To download a feature to a local directory without installing the feature, use the **--downloadOnly** option; for example:

```
bin/featureManager install feature_shortName1 feature_shortName2 --downloadOnly=[all|required*|none]
```

You can configure this option to download all the dependent features, the dependent features required for this runtime, or none of the dependent features. The default is to download the required dependent features. To specify a local destination directory, use this option with the **--location=directoryPath** option.

-  **8.5.5.4** To install features from a local source directory, use the **--location=directoryPath** option; for example:

```
bin/featureManager install feature_shortName1 feature_shortName2 --location=directoryPath
```

-  **8.5.5.4** If you do not want to connect to the Liberty Repository, use the **--offlineOnly** option to install features from a local directory; for example:

```
bin/featureManager install feature_shortName1 feature_shortName2 --offlineOnly --location=directoryPath
```


- Use the **featureManager** command to see what assets you have installed or to get help. For more information, see: “featureManager command”

## featureManager command

You can use the **featureManager** command to find, install, uninstall, or obtain details about features.

The feature that you want to install must be packaged as a subsystem archive (ESA file).

**8.5.5.5** You can access the Liberty Repository using the featureManager command through a proxy server. For more information, see “Configuring proxy server support for the featureManager command in Liberty” on page 871.

**Tip:**  Use the **installUtility** command instead of the **featureManager** command. With the **installUtility** command, you can manage more asset types and install, find, or download assets from multiple repositories. For more information, see “Installing assets using the installUtility command” on page 853.

## Syntax

The command syntax is as follows:

```
featureManager action [options]
```

where *action* can take one of the following values:

**install** Install a feature from an enterprise subsystem archive (ESA) file, a URL to an ESA file, a feature short name, or a feature symbolic name.

If you specify a feature short name or a symbolic name, the feature is downloaded from an online repository hosted by IBM.

The *install* action uses the OSGI-INF/SUBSYSTEM.MF file from the ESA file as a new feature manifest that can be copied into the `${wlp.user.dir}/extensions/lib/features` directory (or product extension directory if a value is specified for the `--to` property) being renamed after the symbolic name of the subsystem. All of the bundles for the subsystem will be extracted into the `${wlp.user.dir}/extensions/lib` directory and renamed as `{bundle symbolic name}_{bundle version}.jar`. License files, checksum files, localization files and other subsystem content will also be extracted to the location defined in the subsystem manifest.



**8.5.5.2**

The **featureManager** command can download assets from the Liberty Repository; for more information, see “Installing Liberty Repository assets” on page 852.

**8.5.5.5**

### uninstall

Uninstall features using either the feature short name or the feature symbolic name.

#### Note:

- Ensure that all server processes are stopped before you uninstall a feature.
- The **uninstall** command cannot uninstall user features; for example: `webCacheMonitor-1.0`.

## featureList

Generate an XML file that contains a report that details all the features that are installed.



**8.5.5.4**

### find

Find assets in the Liberty Repository that are applicable to your configuration.

**8.5.5.4**

### classpath

Generate a JAR file that can be added to a compiler classpath in order to use APIs from a list of features. This action enables you to compile build scripts against the API JARs that are included in the product without referencing specific JAR names, which can change when a fixpack is applied. The output JAR contains relative paths to the API JARs in the product. Therefore, you must not move the output JAR to another directory.

**Note:** The `--features` option must be specified with this action.

**help** Display help information for a specified action.

## Options

The following options are available for the **featureManager install** command:

### --acceptLicense

Automatically indicate acceptance of license terms and conditions.

**8.5.5.4** **--downloadOnly=[all | required\* | none]**

Download the requested feature to a local directory without installing the feature. This option can be configured to download all the dependent features, the dependent features required for this runtime, or none of the dependent features. The default is to download the required dependent features. Specify the directory with the **--location** option.

**Note:** You cannot use this option if you specify the subsystem archive location with a file name or URL.

**8.5.5.4** **--location=directoryPath**

Specifies the location of the subsystem archive that you want to install. When used with the **--downloadOnly** option, this option specifies a destination directory for downloaded features. This option is required when using the **--downloadOnly** and **--offlineOnly** options.

**Note:** You cannot use this option if you specify the subsystem archive location with a file name or URL.

**8.5.5.4** **--offlineOnly**

Use this option if you do not want to connect to the Liberty Repository. Instead, the command only installs features from the local directory. The local directory is specified with the **--location** option.

**Note:** You cannot use this option if you specify the subsystem archive location with a file name or URL.

**--to=install\_option**

where *install\_option* can take one of the following values:

- **usr:** The feature is installed as a user feature. This is the default value.
- **extension:** The location to which you want to install the feature. You can install the feature to any configured product extension location.

**--viewLicenseAgreement**

View license agreement.

**--viewLicenseInfo**

View license information.

**--when-file-exists=exist\_option**

Specifies the action to take if a file to be installed already exists. *exist\_option* can take one of the following values:

- **fail:** Cancel the installation.
- **ignore:** Continue the installation and ignore the file that exists.
- **replace:** Overwrite the existing file.

**--verbose**

Use this option to display any available additional information while the action runs.

**subsystem\_archive**

Specifies the location of the subsystem archive that you want to install. You can specify the location in the following ways:

- A file name; for example: `my_feature.esa`.
- A URL, for example:

`http://myhost.ibm.com/liberty/assets/my_feature.esa`

- **8.5.5.2** The short name of the subsystem archive (ESA file), such as `adminCenter-1.0`
- **8.5.5.2** The symbolic name of the subsystem archive (ESA file), such as `com.ibm.websphere.appserver.adminCenter-1.0`

For more information, see the Downloads page on WASdev.net.

**Note:** Specify multiple features by separating the features with a comma or a space. For versions prior to Version 8.5.5.7, multiple features must be separated by a comma.

8.5.5.5 The following options are available for the **featureManager uninstall** command:

8.5.5.8 **--force**

Uninstall the specified feature regardless of whether other installed features have dependencies on it.

**Note:** Uninstalling a feature that is required by other installed features might cause those features to stop working and might prevent servers from running correctly.

**--noPrompts**

Uninstall features without any user interaction or confirmation messages.

**--verbose**

Use this option to display any available additional information while the action runs.

**name** Specify the feature to uninstall. You can specify the following options:

- The short name of the subsystem archive (ESA file), such as adminCenter-1.0.
- The symbolic name of the subsystem archive (ESA file), such as com.ibm.websphere.appserver.adminCenter-1.0.

**Note:** Specify multiple features by separating the features with a comma or a space. For versions prior to Version 8.5.5.7, multiple features must be separated by a comma.

The following options are available for the **featureManager featureList** command:

**--encoding=charset**

where *charset* is the character set to use when creating the XML report file.

**--locale=language**

where *language* specifies the language to use when creating the XML report file. This consists of the ISO-639 two-letter lowercase language code, optionally followed by an underscore and the ISO-3166 uppercase two-letter country code.

**--productExtension=name**

where *name* is the product extension name whose features are to be listed. If the product extension is installed in the default user location, use the keyword: *usr*. If this option is not specified, the action is taken on WebSphere Application Server Liberty Core.

**XML\_report\_file\_name**

Specifies the name of the XML report file that you want to create.

The following options are available for the **featureManager find** command:

8.5.5.4 **--viewInfo**

View detailed information.

**--verbose**

Use this option to display any available additional information while the action runs.

8.5.5.4 **searchString**

Use the *searchstring* option to find applicable features from the IBM WebSphere Liberty Repository

8.5.5.4 The following options are available for the **featureManager classpath** command:

`--features=feature1,feature2,...`

The list of features that contain the list of API JAR files. This option is required for the classpath action.

*fileName*

The name of the generated JAR file.

## Usage examples

The following example installs the subsystem archive `my_feature.esa` as a user feature:

```
featureManager install my_feature.esa --to=usr
```

The following example installs the subsystem archive `my_feature.esa` to the `my_extension` product extension location:

```
featureManager install my_feature.esa --to=my_extension
```

The following example generates a report for all installed features; the report is written to the file `my_feature_report.xml` using the Brazilian Portuguese language:

```
featureManager featureList my_feature_report.xml --locale=pt_BR
```

The following example generates a report that contains all installed features that are defined in the product extension that is installed in the default user location `wlp/usr/extension`, which is known by the runtime environment as the `usr` product extension. The report is written to the file `my_feature_report.xml`:

```
featureManager featureList --productExtension=usr my_feature_report.xml
```

The following example generates a report that contains all features that are defined in the product extension that is installed in the location pointed to by the content in the `productExtensionName.properties` file in the product installation's `etc/extensions` directory. The report is written to the file `my_feature_report.xml`:

```
featureManager featureList --productExtension=productExtensionName my_feature_report.xml
```

The following example displays help information for the `install` action:

```
featureManager help install
```



8.5.5.4

The following example finds applicable assets from the Liberty Repository:

```
featureManager find searchString
```



8.5.5.4

The following example lists detailed information:

```
featureManager find searchString --viewInfo
```



8.5.5.4

The following example downloads required dependencies from the Liberty Repository to a local directory and does not install them:

```
featureManager install feature_shortName --downloadOnly --location=c:\temp\download --acceptLicense
```

**Note:** You cannot use this option if you specify the subsystem archive location with a URL.



8.5.5.4

The following example downloads all dependencies from the Liberty Repository and does not install them:

```
featureManager install feature_shortName --downloadOnly=all --location=c:\temp\download --acceptLicense
```

**Note:** You cannot use this option if you specify the subsystem archive location with a URL.



8.5.5.4

The following example installs features that are located in a local directory. If there are missing dependencies, they are installed from the online repository:

```
featureManager install feature_shortName --location=c:\temp\download --acceptLicense
```

**Note:** You cannot use this option if you specify the subsystem archive location with a URL.



8.5.5.4

The following example installs assets from a local directory without downloading missing dependencies from the Liberty Repository:

```
featureManager install feature_shortName --location=c:\temp\download --offlineOnly
```

**Note:** You cannot use this option if you specify the subsystem archive location with a URL.



8.5.5.4

The following example creates a classpath JAR file and compiles an application class that uses it:

```
featureManager classpath --features=servlet-3.0 classpath.jar
javac -cp classpath.jar TestServlet.java
```



8.5.5.5

The following example uninstalls a feature:

```
featureManager uninstall adminCenter-1.0
```

## Return codes

Table 69. Return codes and explanations

Return code	Explanation
0	The command successfully completed the requested operation.
20	One or more arguments are not valid.
21	A runtime exception occurred because of one or more of the following conditions: <ul style="list-style-type: none"> <li>A runtime exception occurred while installing the .esa subsystem archive file.</li> <li>A license is not accepted or acknowledged.</li> <li>The .esa subsystem archive file did not extract correctly.</li> </ul>
22	The feature that you wanted to install already exists.
23	The feature definition was not valid for one or more of the following reasons: <ul style="list-style-type: none"> <li>The feature does not have a valid manifest file.</li> <li>The version of the feature is not supported in this Liberty environment.</li> <li>The .zip or .jar file that contains the feature files does not exist.</li> </ul>
24	The .esa subsystem archive file is missing content.
25	A file that you wanted to install already exists, and you specified the <b>when-file-exists=fail</b> option.
26	The product is not a core product, and the product extension files cannot be found.
27	The product is not a core product, and the product extension is not defined in the <code>\${wlp.install.dir}/etc/extensions/<i>extension_name</i>.properties</code> file.

Table 69. Return codes and explanations (continued)

Return code	Explanation
28	The manifest files for the feature in the product extension cannot be found.
29	The feature is not valid for the current product.

## Configuring proxy server support for the featureManager command in Liberty

8.5.5.5

You can access the Liberty Repository using a proxy server.

### Before you begin

Ensure that you have created the required directory structure. If the etc directory is not found in the Liberty installation directory, then you must create it.

### About this task

You can run the **featureManager** command to connect to the IBM WebSphere Liberty Repository through a user-specified proxy server.

#### Note:

- The **featureManager** command only supports proxy servers with HTTP/HTTPS protocols.
- The proxy server is only supported for the `install` and `find` options for the **featureManager** command.
- To access the IBM WebSphere Liberty Repository with limited internet access or through a firewall, ensure that you have access to the following hosts and ports:
  - `public.dhe.ibm.com` on port 443
  - `asset-websphere.ibm.com` on port 443

8.5.5.6

Both the **featureManager** command and the **installUtility** command use the same configuration properties file, `repositories.properties`. The proxy settings are shared, but the repository settings are only used by the **installUtility** command. Using the **installUtility** command is recommended over the **featureManager** command because it enables you to work with more asset types and from multiple repositories; for more information see “Configuring repositories and proxy settings for the `installUtility` command” on page 862.

To configure proxy server support for the **featureManager** command, complete the following steps.

### Procedure

1. In the `<Liberty_installation_dir>/etc` directory, create an ASCII properties file, and name it `repositories.properties`.
2. In the `repositories.properties` file, specify the following properties with the proxy server configurations:
  - a. Specify the proxy server host name with no protocol information.  
`proxyHost=proxy.xyz.com`
  - b. Specify the proxy server port number:  
`proxyPort=8080`
  - c. Specify the proxy server user name and encrypted password. The proxy server password must be encrypted by using the **securityUtility** command found in the `<Liberty_installation_dir>/bin/` directory.

For more information about the encoding options, see: “securityUtility command” on page 1162.If the proxy server does not require authentication, then the **proxyUser** and **proxyUserPassword** properties are not required.

```
proxyUser=user
proxyUserPassword={aes}Ly0wJyYPKDs=
```

3. Save the `repositories.properties` file.

## What to do next

Run the `featureManager install name` or `featureManager find searchString --viewInfo` commands to install or query applicable Liberty features from the IBM Liberty Repository through the specified proxy server. For more information, see “featureManager command” on page 865.

## Installing assets using Installation Manager



If you have IBM Installation Manager installed, you can download and install Liberty Repository assets when you install or upgrade Liberty.



### Before you begin

**Note:** To install Liberty Repository assets, you must have IBM Installation Manager Version 1.6.2 or later and be installing or upgrading to Liberty for WebSphere Application Server Version 8.5.5.2 or later.

### About this task

This task applies only to installing Liberty Repository features or add-ons during the installation or upgrade of Liberty for WebSphere Application Server Version 8.5.5.2 or later. For information about installing product samples and open source integrations, see Installation Manager repository for Liberty samples on WASdev.net. For a list of all assets and their descriptions, see the Downloads page on WASdev.net.

You can install Liberty Repository assets from the following repositories:

- The IBM WebSphere Liberty Repository, a public IBM-hosted repository that is accessible through the internet. For more information, see “Liberty Repository” on page 573.
-  The Liberty Asset Repository Service, an open-source service that you can use to create an on-premises repository that is remotely accessible behind the firewall of an enterprise. For more information, see the WASdev/tool.lars project on GitHub.
-  Local directory-based repositories that are created by using the **installUtility download** action. For more information, see “Downloading assets using the installUtility command” on page 860.

Rather than downloading individual assets, you can download a `wlp-featureRepo-<version>.zip` file from IBM Fix Central. The `.zip` file contains a directory-based repository of all features and add-ons for that fix pack. Downloading assets using these methods creates a `repository.config` file in the repository that you must point to from Installation Manager.

You can add the directory-based and Liberty Asset Repository Service repositories to Installation Manager in the same manner that you add the product offering repositories, then install assets from the repositories. Assets are installed from the repositories in the order that you specify them. If these repositories do not contain the assets that you want to install, the assets are installed from the IBM WebSphere Liberty Repository unless you disable it.

To access the IBM WebSphere Liberty Repository with limited internet access or through a firewall, ensure that you have access to the following hosts and ports:



- public.dhe.ibm.com on port 443
- asset-websphere.ibm.com on port 443

## Procedure

1. **8.5.5.6** Optional: Set up an instance of the Liberty Asset Repository Service or a local directory-based repository, then specify the repository URL or directory path in Installation Manager.
  - 8.5.5.8** For Version 8.5.5.8 and later, you can also specify the file path of a directory-based repository compressed in an archive file.
2. Install Liberty and Liberty Repository assets.
  - **Distributed operating systems** Choose from the following options for working with Installation Manager:
    - Graphical User Interface (GUI) - The Installation Manager GUI provides a list of available assets and detailed descriptions of each asset. See “Installing Liberty on distributed operating systems using the GUI” on page 697.
    - Command line - The **imcl** command can download and install assets. See “Installing Liberty on distributed operating systems by using the command line” on page 700.
    - Response files - Installation Manager supports installing assets using recorded or manually created response files. See “Installing Liberty on distributed operating systems by using response files” on page 703.
  - **IBM i** Choose from the following options for working with Installation Manager:
    - Command line - The **imcl** command can download and install assets. See “Installing Liberty on IBM i operating systems using the command line” on page 790.
    - Response files - Installation Manager supports installing assets using recorded or manually created response files. For detailed steps about installing assets using Installation Manager in silent mode, see “Installing Liberty on IBM i operating systems using response files” on page 794.

## Installing assets by using developer tools

**Distributed operating systems**  **8.5.5.2**

If you have developer tools installed, you can install feature assets, open source integration assets, product sample assets, runtime assets, add-on assets, and config snippet assets from the Liberty Repository.

### About this task

Developer tools for WebSphere Application Server, such as IBM WebSphere Application Server Developer Tools for Eclipse, provide a graphical user interface that you can use to install Liberty Repository assets. This task shows you how to install assets by using WebSphere Developer Tools, Version 8.5.5.2 and later versions.

## Procedure

- Optional: Install assets from the Servers view.
  - | If you create a Liberty server and a Liberty runtime environment, either from an archive or from the Liberty repository, you can install the following assets on the new runtime environment in the Servers view:
    - | – Feature assets
    - | – Open source integration assets
    - | – Product runtime environment assets
    - | – Product sample assets

- | – Add-on assets
- | You can install configuration snippet assets after you create a Liberty server. For more information, see
- | “Creating a Liberty server by using developer tools” on page 884.

- Install assets from the Runtime Explorer view.

In the Runtime Explorer view, you can install the following assets when you create a runtime environment or when you create a server in a runtime environment:

- Feature assets
- Open source integration assets
- Product sample assets
- Add-on assets

For more information, see the “Exploring the runtime environment by using developer tools” on page 943 topic.

- Review the assets on the Downloads page on WASdev.net.

You can use search and filters to narrow your search results. Assets that you can install by using WebSphere Developer Tools are included on the site, along with assets that you can download by other methods.

## Adding more repositories by using developer tools

8.5.5.7

You can configure the list of repositories that are available for access by the tools. You can add remote repositories or local, directory-based repositories.

### Before you begin

- To create an on-premises repository that is remotely accessible behind the firewall of an enterprise, see the WASdev/tool.lars project on GitHub.
- To create a local, directory-based repository by using the `installUtility` download action, see Downloading assets using the `installUtility` command.

### Procedure

1. Open the Runtime Explorer view.
  - a. Select **Window > Show View > Other** from the workbench toolbar.
  - b. Select **Server > Runtime Explorer** in the Show View window.
2. Right-click the runtime environment, and then select **Install additional content...**  
The Install Additional Content window is displayed.
3. Click **Configure Repositories**.  
The Configure Repositories window is displayed.
4. Add a repository.
  - a. To add a repository, click **Add...**
  - b. Enter a name for the repository in the **Name** field.
  - c. Enter the location for the repository.
    - To use a remote repository, select **Remote Repository**, and then enter the connection information.
    - To use a repository in a local directory, select **Local Repository**. Then, enter or browse to the location of the assets, which you downloaded by using the `installUtility` command of the runtime environment, or from the Liberty Repository.
    - **8.5.5.8** To use a repository in a .zip file, select **Browse**, and then browse to your .zip file. You can download a `wlp-featureRepo-<version>.zip` file from IBM Fix Central. The .zip file contains a directory-based repository of all features and add-ons for that fix pack.

- d. Click OK.

## What to do next

In the Configure Repositories window, you can add, edit, remove, and reorder repositories.

The repository list is in descending order of priority. If an asset is contained in more than one repository, then the repository that is found first in the list is used during installation. Ensure that your repository is selected.

---

## Verifying the integrity of Liberty installation

You can use the command utilities to verify the installation integrity of Liberty.

### About this task

After you have installed Liberty, you must make sure that the installation is completed successfully and that all required features or iFixes are installed. The `productInfo` command provides different options to complete the following tasks:

- Compare the differences between APAR fixes and the current installation.
- Validate the MD5 checksum file for server installation and each feature.
- Verify the version information of the current installation.
- Verify the feature list on the current installation.

### productInfo command

Use the `productInfo` command to validate the product integrity, compare different versions of the Liberty servers, and verify the current product versions.

### Syntax

The command syntax is as follows:

```
productInfo action --[options]
```

Where the possible values of the *options* vary depending on the value of the *action* parameter.

### Parameters

The following *action* parameters and *options* values are available for the `productInfo` command:

#### compare

Allows you to compare APAR fixes that are installed in the current installation with a different version of Liberty.

**--target**=*path to directory or archive file*

Specifies the target file with which to compare the current installation. The value of **--target** can be either a directory or an archive file that must be a valid Liberty installation location. This option is required if **--apars** is not specified.

**--apars**=*a comma separated list of APAR IDs*

Checks the current installation against this comma-separated list of APAR IDs to see if it contains them, and then lists any APARs that are not included. This option is required if **--target** is not specified.

**--output**=*path to an output file*

Outputs the result from this command to the supplied file name. By default, the output is directed to standard output.

**--verbose**

Displays detailed error messages when an error occurs.

**Note:** At least one of **--target** or **--apars** must be supplied

**featureInfo**

Lists all the features that are installed on the current Liberty server including any installed product extensions.

**--output=***path to an output file*

Outputs the result from this command to the supplied file name. By default, the output is directed to standard output.

**validate**

Validates the Liberty server.

**--output=***path to an output file*

Outputs the result from this command to the supplied file name. By default, the output is directed to standard output.

**version**

Displays the product name and version.

The output includes the name and version of product extensions if a *product\_extension.properties* file is provided in the product extension installations versions directory containing the following properties:

`com.ibm.websphere.productVersion=your_product_version.`

`com.ibm.websphere.productName=your_product_name.`

**--output=***filename*

Outputs the result from this command to the supplied file name. By default, the output is directed to standard output.

**--verbose**

Displays the whole content of each properties file.

**--ifixes**

Displays the APAR fixes that are applied to the system, and the interim fixes that applied them.

For more information, refer to “Provide product information for your feature extension” on page 1143

**8.5.5.6 viewLicenseAgreement**

Displays the license agreement for the Liberty edition that is installed.

**8.5.5.6 viewLicenseInfo**

Displays the license information for the Liberty edition that is installed.

**help** Displays help information for the specific action.

**Usage**

The following examples demonstrate correct syntax: **8.5.5.6**

```
productInfo compare --target=C:\wlp\newInstall\wlp
productInfo compare --target=C:\wlp\newInstall.jar --output=C:\wlp\compareOutput.txt
productInfo compare --apars=com.ibm.ws.apar.PM39074,com.ibm.ws.apar.PM39075,com.ibm.ws.apar.PM39080
productInfo featureInfo --output=c:\wlp\featureListOutput.txt
productInfo validate
```

productInfo help compare  
productInfo version  
productInfo viewLicenseAgreement  
productInfo viewLicenseInfo

---

## Docker support in Liberty

8.5.5.5

Docker is an open source platform that uses Linux containerization and a layered file system. The image that is built by using Docker consists of layers that contain the application and the dependent binary files and libraries, but uses the kernel from the host operating system. The layering allows changes to be made quickly because only the modified layers are rebuilt and deployed.

Liberty is designed to run in a container on multiple platforms compatible with the Docker engine. Specifically, Liberty has been tested on IBM Containers, Docker Datacenter, and OpenShift V3. To qualify for support, the base operating system that is used within the container must be a platform supported by the Liberty product.

For more information about Docker and its advantages, see [What is Docker?](#)

Docker Hub is a public repository that hosts Docker images. A Docker image for WebSphere Application Server for Developers Liberty is available on Docker Hub along with the documentation that describes its usage.

Running Liberty under Docker is also supported for production usage. Instructions and Dockerfiles for building production licensed Docker images by using the installation files that are obtained from Passport Advantage can be found on WASdev GitHub.

---

## Accessing a remote Liberty server in a Docker container by using developer tools

8.5.5.8

You can set up your remote Liberty server in a Docker container so that you can access it by using WebSphere Developer Tools. After you complete this setup, you can use WebSphere Developer Tools to configure and start your remote Liberty server.

### Before you begin

1. Install Docker, Cloud Foundry Command Line Interface (CLI), and the Cloud Foundry plug-in for IBM Containers.

For more information about installation, see [IBM Containers plug-in](#).

2. Log in to your Bluemix account, choose your organization and space, and then log in to your IBM Containers service.

For more information about logging in to your accounts, see [Logging into the CLI](#).

**Tip:** You can either run `cf ic` commands or `docker` commands to set up a Liberty server in a Docker container. For more information about `cf ic` commands and `docker` commands, see the Step 5 section of [Logging into the CLI](#).

### Procedure

1. To access your remote Liberty server in a Docker container, use the following command:

```
docker exec -i ContainerID LibertyInstallDirectory/bin configUtility install remoteAdministration --vadminUser=Userm
```

**Remember:** To find the container ID, use the `docker ps` command. By default, the `KeystorePassword` value is `Liberty`.

Also, by default the `LibertyInstallDirectory` value is `/opt/ibm/wlp`, and the `bin` directory is in the `/opt/ibm/wlp` directory.

For more information about the `configUtility` command, see [Liberty: configUtility command](#).

2. When the script displays the configuration snippet, copy it into the `remoteAdministration.xml` file. The `remoteAdministration.xml` file is in the `LibertyInstallDirectory/usr/servers/serverName/configDropins/defaults` directory. If this directory and file do not exist, create the directory and then create the file.
3. Specify the `<remoteFileAccess>` parameter in the `remoteAdministration.xml` file by replacing its contents with the following text:

```
<writeDir>${server.config.dir}</writeDir>
<writeDir>${server.output.dir}</writeDir>
<writeDir>${wlp.user.dir}</writeDir>
```
4. Save the `remoteAdministration.xml` file.

## What to do next

You can configure and start a remote Liberty server that is in a Docker container.

For more information about how to configure and start a remote Liberty server, see “Creating a remote Liberty server by using developer tools” on page 887.

---

## Liberty and Chef

Chef software is an open source configuration management tool that you can use to create and manage the installation of an Infrastructure as a Service (IaaS). You can use Chef to provision a Liberty installation.

Chef uses cookbooks, which are reusable sets of components that are written in the Ruby programming language. A cookbook provides all the necessary components that are needed to configure an associated piece of software, for example, Apache HTTP Server. An important distinction between using Chef and writing scripts is the ability of Chef to determine the differences between the current software configuration and a new configuration and make only the changes necessary to move from one to the other. For more information, see the online Chef documentation [About Cookbooks](#).

By using Chef, you can create scalable infrastructure with minimal configuration to maintain. You can easily expand your existing infrastructure, for example, by creating and starting a new application server or web server. Chef automatically connects the new servers into the existing infrastructure as necessary.

The `wlp` cookbook installs and configures the WebSphere Application Server Liberty. It provides recipes, resources, and libraries for creating, managing, and configuring Liberty server instances. For more information, see [wlp cookbook](#).

To learn more about Chef, see the online Chef documentation [How Chef works](#).

To learn more about using Chef cookbooks, see [Getting started with the Chef cookbooks for Liberty](#).

---

## Installing the OpenShift Cartridge for Liberty

Distributed operating systems

IBM i

8.5.5.9

The downloadable OpenShift cartridge allows the Liberty server to be available on OpenShift.

## Before you begin

- You must have Ruby version 1.9.3 or later installed.
- This document details how to download the OpenShift cartridge by using the Linux command line.

## About this task

OpenShift is an open source platform that allows developers to quickly develop, build, deploy, and manage Linux containerized services and applications in a cloud environment. When you install the OpenShift cartridge for Liberty, both Liberty and JRE binary files are downloaded for each application.

The OpenShift cartridge is also available at <https://github.com/WASdev/cloud.openshift.cartridge.wlp> where you can find extra documentation.

For more information about OpenShift and its advantages, see OpenShift

**Note:** Only OpenShift version 2 can be used with this cartridge

## Procedure

Installing the cartridge into your OpenShift environment

1. Download the cartridge compressed file from IBM Fix Central.
2. Run the **unzip** command to extract the contents of the cartridge `ibm-websphere-liberty-cartridge-v*.zip` file to the following cartridge directory `cloud.openshift.cartridge.wlp`
3. The OpenShift cartridge downloads the default Liberty and IBM JRE binary files for development only. If you have licenses for other binary files that you want to use, they need to be accessible with HTTP. See for details on how to structure this repository.
  - a. Update the `ibm-websphere-liberty-buildpack/config/ibmjdk.yml` file to point to your JRE binary.
  - b. Update the `ibm-websphere-liberty-buildpack/config/liberty.yml` file to point to your Liberty binary.
4. Run the **chmod u+x** `ibm-websphere-liberty-buildpack/resources/download_buildpack_cache.rb` command and add the binary files from step 3 to the cartridge cache. Run the **ibm-websphere-liberty-buildpack/resources/download\_buildpack\_cache.rb** `ibm-websphere-liberty-buildpack/admin_cache`
5. If you are licensed to deploy the cartridge into your environment, you can create a `ibm-websphere-liberty-buildpack/config/licenses.yml` file that contains the accepted license numbers before packaging.

```
IBM_JVM_LICENSE: <jvm license code>
IBM_LIBERTY_LICENSE: <liberty license code>
```

**Note:** When installed the cartridge can be made available to all users. If you are an administrator or own your OpenShift installation, you can have more control over how the cartridge behaves in these situations. After you add the license to the cartridge package, individual applications do not need to accept the license terms with environment variables.

6. Return to your original directory and copy the cartridge directory to each node host on your OpenShift machines.
7. On each node host run:

```
oo-admin-cartridge --action install --source cloud.openshift.cartridge.wlp/
service ruby193-mcollective restart
```
8. On the broker host run:

```
oo-admin-broker-cache --clear --console
oo-admin-ctl-cartridge -c import-node --activate
oo-admin-console-cache --clear
```

9. Verify that the cartridge is installed by running the **rhc cartridges** command.  
Uninstalling the cartridge from your OpenShift environment
10. On the broker host run:  

```
oo-admin-ctl-cartridge -c deactivate --name ibm-liberty-8.5.5
```
11. On each node host run:  

```
oo-admin-cartridge --action erase --name liberty --version 8.5.5 --cartridge_version <Cart_Version_Number>
```
12. On the broker host run:  

```
oo-admin-broker-cache --clear --console
oo-admin-console-cache --clear
```
13. Verify that the cartridge is uninstalled by running the **rhc cartridges** command.

## Results

The OpenShift cartridge is now installed.

---

## Installing the IBM WebSphere Application Server Liberty Buildpack into a Cloud Foundry Environment

Distributed operating systems

IBM i

8.5.5.9

The IBM WebSphere Application Server Liberty Buildpack makes the Liberty server available in Cloud Foundry.

### Before you begin

- You must have Ruby version 1.9.3 or later installed.
- This document details how to download the IBM WebSphere Application Server Liberty buildpack by using the Linux command line.

### About this task

Use this task to install the IBM WebSphere Application Server Liberty Buildpack into a Cloud Foundry Environment. If you are a Cloud Foundry administrator, you can install the Liberty buildpack as an admin buildpack making it available to all users within the Cloud Foundry.

**Note:** Buildpack users do not need to specify the `-b` option to use the buildpack directly from the administrator.

The IBM WebSphere Application Server Liberty Buildpack is also available at <https://github.com/cloudfoundry/ibm-websphere-liberty-buildpack> where you can find extra documentation.

## Procedure

Installing the buildpack into your Cloud Foundry environment

1. Download the buildpack compressed file from IBM Fix Central.
2. Run the **unzip** command to extract the contents of the buildpack `ibm-websphere-liberty-buildpack-v*.zip` file to the following buildpack directory `ibm-websphere-liberty-buildpack`
3. The IBM WebSphere Application Server Liberty buildpack downloads the default Liberty and IBM JRE binary files for development only. If you have licenses for other binary files that you want to use, they need to be accessible with HTTP. See for details on how to structure this repository.
  - a. Update the `config/ibmjdk.yml` file to point to your JRE binary.
  - b. Update the `config/liberty.yml` file to point to your Liberty binary.



4. If you are licensed to deploy the buildpack into your environment, you can create a `config/licenses.yml` file that contains the accepted license numbers before packaging.

```
IBM_JVM_LICENSE: <jvm license code>
IBM_LIBERTY_LICENSE: <liberty license code>
```

**Note:** After you add the license to the buildpack package, individual applications do not need to accept the license terms with environment variables.

5. Install Ruby gems that are needed to package an admin buildpack by running the **gem install bundler** and **bundle install** commands.
6. Run the **bundle exec rake** package task to create an admin buildpack.

```
rake 'package[zipfile,hosts,version]'
```

The **zipfile** parameter is the name of the generated admin buildpack and includes a relative location that is NOT the current directory. For example, `../my-admin-buildpack.zip` can be specified as the **zipfile** parameter to generate the `my-admin-buildpack.zip` file in the parent directory instead of the default `ibm-websphere-liberty-buildpack-480d2de.zip` file.

For example,

```
rake 'package[../my-admin-buildpack.zip]'
```

The **hosts** parameter is a list of sites that the package task pulls binary files from for inclusion in the admin buildpack. By default, only binary files from the public IBM site are pulled. IBM hosted sites do not include third-party binary files. A package parameter must be specified to indicate that third-party binary files can be included in the admin buildpack for cases where the admin buildpack is used in offline mode. Using `*` includes all the binary files in the admin buildpack if the download is possible during the packaging.

An example of this usage:

```
rake 'package[,*,]'
```

The **version** parameter is the version information that is displayed when an application is deployed to **CloudFoundry** with the Cloud Foundry command line interface. By default, the displayed version is the latest commit identifier, such as `480d2de`.

The following example illustrates the displayed version information defaults.

```
Liberty Buildpack Version: 480d2de | git@github.com:cloudfoundry/ibm-websphere-liberty-buildpack.git#480d2de
```

7. Install the admin buildpack with the `cf` client as follows:

```
cf create-buildpack ibm-websphere-liberty-buildpack ibm-websphere-liberty-buildpack-480d2de.zip 1
```

- `ibm-websphere-liberty-buildpack` is the name that is given to the admin buildpack.
- `ibm-websphere-liberty-buildpack-480d2de.zip` is the path to the compressed file that is created by the Rake task.
- `1` is the priority given to the admin buildpack. The lower the number, the higher the priority.

See the Cloud Foundry documentation for further details.

## Results

The IBM WebSphere Application Server Liberty Buildpack is now installed.



---

## Chapter 4. Setting up Liberty

Define directory locations and variables, create and configure servers, and add and remove Liberty features that specify the capabilities of your server.

### Procedure

- Defining directory locations and properties.
- “Verifying the integrity of Liberty installation” on page 875.
- **Distributed operating systems** “Creating a Liberty server by using developer tools” on page 884.
- “Creating a Liberty server manually.”
- “Specifying Liberty bootstrap properties” on page 897.

The default HTTP port is 9080 and HTTPS port is 9443 for Liberty. You can manually assign appropriate port numbers in the `server.xml` files when multiple Liberty servers are running on the same machine.

---

## Creating a Liberty server manually

You can create a server from the command line.

### **Distributed operating systems**

### Before you begin

### **Distributed operating systems**

You can create a server as described here, or as described in “Creating a Liberty server by using developer tools” on page 884.

### Procedure

1. Open a command line, then change directory to the `wlp/bin` directory.  
Where `path_to_liberty` is the location you installed Liberty on your operating system.  
Example on Windows: `C:\Users\mo> cd path_to_liberty\wlp\bin`  
Example on Linux: `mo@machine01:~> cd path_to_liberty/wlp/bin`
2. Run the following command to create a server. If you do not specify a server name, `defaultServer` is used.

Where `server_name` is the name you want to give your server.

Example on Windows: `C:\wlp\bin> server create server_name`

Example on Linux: `mo@machine01:~> server create server_name`

Windows

AIX

Linux

UNIX

HP-UX

Solaris

IBM i

```
server create server_name
```

`server_name` must use only Unicode alphanumeric (for example, 0-9, a-z, A-Z), underscore (`_`), dash (`-`), plus (`+`), and period (`.`) characters. The name cannot begin with a dash or period. Your file system, operating system, or compressed file directory might impose additional restrictions.

### Results

If the server is created successfully, you receive message: Server `server_name` created.

If the specified server already exists, no server is created and you receive an exception message:

CWWKE0005E: The runtime environment could not be launched.  
CWWKE0045E: It was not possible to create the server called *server\_name* because the server directory C:\wlp\usr\servers\*server\_name* already exists.

A directory with the name of the new server is created under the  $\{\text{wlp.user.dir}\}/\text{servers}$  directory, containing the configuration of the new server. The HTTP port numbers for the new server are assigned to default values and are shown in the generated *server.xml* file to make it easy to edit them. You can also set these values by using variables in a *bootstrap.properties* file in the same directory. For more information, see “Specifying Liberty bootstrap properties” on page 897.

## What to do next

Configure your server to have the features that your application requires. See “Configuring the Liberty runtime environment ” in the “Administering” book.

---

## Creating a Liberty server by using developer tools

### Distributed operating systems

You can use developer tools to create and start a Liberty server. If you have not yet installed Liberty, the developer tools can install it for you when you create a server for the first time.

### Before you begin

Make sure that you installed the developer tools as described in “Installing Liberty developer tools and (optionally) Liberty” on page 835.

You can create a server as described in this topic, or as described in “Creating a Liberty server manually” on page 883.

When you create a new server using the tools, you specify the installation of Liberty that you want to use. You are offered three options:

- **Select an existing installation.**
- **Install from a previously downloaded archive file.**
- For the no-charge developer edition, **Download and install.**

If you want to use the tools to install a Liberty edition (other than the no-charge developer edition) from an archive file, make sure that you have downloaded the archive file.



8.5.5.5

If you need to use a proxy server to connect to the Liberty repository, first configure the proxy settings by selecting **Window > Preferences > General > Network Connections** from the main menu in Eclipse. Enter the information there.

### About this task



Complete the following steps to create and start a Liberty server.

As you go through the steps, you can download add-ons in the Liberty Repository from the WASdev community download site.

### Procedure

1. In the workbench, open the Servers view by clicking the **Servers** tab.




**Tip:** If the Servers view is not visible, navigate to **Window > Show view > Other...** and type Server in the filter text. Then, select **Servers**.

2. Right-click the **Servers** view and select **New > Server**.
3. Under the server type list, expand **IBM** and select the  **8.5.5.4 WebSphere Application Server Liberty** server type.  **8.5.5.4** The WebSphere Application Server Liberty server type replaces the old WebSphere Application V8.5 Liberty server type and supports all versions of the WebSphere Application Server Liberty server. Existing servers in a workspace that were created with the old server type still work.
4. Click **Next**. The Liberty Runtime Environment page is displayed.
- 5.

**Tip:** If you already have a Liberty run time that is installed, you will go directly to the **New Remote Liberty Server** page, skip to Step 7.





Select an installation, install from an archive file, or (for the no-charge developer edition) download and install, Liberty.

If you previously installed Liberty, complete the following steps:

- a.  **8.5.5.4** Select **Choose an existing installation**.
- b.  **8.5.5.4** In the **Path** field, type or browse for the directory where you installed the Liberty runtime environment.
- c.  **8.5.5.4** On the Liberty Runtime Environment page, click **Next**.

The application-serving environment is selected, and now you can skip to Step 7.

If you want to install Liberty from an archive file that was previously downloaded, complete the following steps:


- a.  **8.5.5.4** Select **Install from an archive or repository**, and click **Next**.
- b.  **8.5.5.4** In the **Destination** field, type or browse for the directory where you want to install the Liberty runtime environment.  
If you type a path that does not exist, then a folder for that path is created automatically at the end of Step 10.
- c.  **8.5.5.4** Select **Install a new runtime environment from an archive**.
- d.  **8.5.5.4** In the **Path** field, either type or browse to the archive file on the local file system, and click **Next**.

- e. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, click **Add Archive**. In the **Add-on archive** field, type or browse for the add-on archive file on the local file system, and then click **OK**. If you have more add-on archive files to install from your local file system, repeat this step until you are done.




 **8.5.5.6** If you want to install add-on files from a custom repository, first add the repository by clicking **Configure Repositories**. Click **New...** to add the repository. The files are then added to the list of add-on files.

After you complete your selection in the Install Add-ons page, click **Next**.

- f. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

Now you can skip to Step 7.

If you want to download and install the no-charge developer edition of Liberty, complete the following steps:

- a.  **8.5.5.4** Select **Install from an archive or repository**, and click **Next**.
- b.  **8.5.5.4** In the **Destination** field, type or browse for the directory where you want to install the Liberty runtime environment.  
If you type a path that does not exist, then a folder for that path is created automatically at the end of Step 10.
- c.  **8.5.5.4** Select **Download and install a new runtime environment from ibm.com**, choose a runtime environment version, and then click **Next**.
- d. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, click **Add Archive**. In the **Add-on archive** field, type or browse for the add-on archive file on the local file system, and then click **OK**. If you have more add-on archive files to install from your local file system, repeat this step until you are done.

If you want to install add-on files from a custom repository, first add the repository by clicking **Configure Repositories**. Click **New...** to add the repository. The files are then added to the list of add-on files.

After you complete your selection in the Install Add-ons page, click **Next**.

- e. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

Now you can skip to Step 7.

6. If the Liberty Server page displays, in the **Liberty server** field, use the drop-down list to select an existing server. Or click **New** to create a new server.

**Note:** This step is skipped and the New Liberty Server dialog is displayed directly if there are no defined Liberty Servers to choose from.

7. If you are creating a new server, in the **Server name** field of the New Liberty Server page, enter a server name of your choice or use the default server name, defaultServer. Then, click **Next** if available, otherwise click **Finish**.
8. In the Liberty Server, click **Next**.
9. Optional: Add the projects of your application to the server. On the **Add and Remove** page, under the **Available** list, select the projects that you want to add to the server and click **Add**. The project appears in the **Configured** list.
10. Click **Finish**.

## What to do next

- Edit the server configuration. For more information, see topic “Editing the Liberty configuration by using developer tools” in the “Administering” book.

- Start the server in “start” mode or in “debug” mode, stop the server, add or remove applications on the server, and many other tasks. You can perform these tasks by using the server context menu (right-click on the server to open the pop-up menu) or by selecting the tray buttons in the Servers view.

**Tip:** In the Servers view, you must select the server entry to perform these tasks. Do not select the server configuration, such as the Server Configuration [server.xml] entry for performing these tasks.

- Optionally configure your server to do specific tasks such as configure Liberty to authenticate users with Tivoli® Directory Server: Right-click **Servers**. Select **Utilities**> **Add config snippets**. After you select the snippets and accept any licenses, the selected configuration snippets are then downloaded and included in the **server.xml** file.

---

## Creating a remote Liberty server by using developer tools

Distributed operating systems



8.5.5.4

You can use developer tools to create and start a remote Liberty server.

### Before you begin

You must meet prerequisites for your local system and your remote system.

- The local system refers to the system where you installed the developer tools.
- The remote system refers to the system where you have Liberty runtime environment that is installed and a Liberty server created.

For your local system, ensure that you meet these prerequisites:

1. The developer tools are installed.  
For more information, see “Installing Liberty developer tools and (optionally) Liberty” on page 835.
2. The Liberty runtime environment is installed. For more information, see “Installing and uninstalling Liberty using downloaded files and archives” on page 833.

For your remote system, ensure that you call the `configUtility` to download and set up the `remoteAdministration` snippet from the repository. Copy the config text that is retrieved by the `configUtility` into the `server.xml` file. For more information, see “`configUtility` command” on page 957.

You can use the following sample remote configuration as an example for your `server.xml` file:

```
<server description="new server">
<!-- Enable features-->
<featureManager>
<feature>restConnector-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" password="password" />

<quickStartSecurity userName="admin" userPassword="password"/>

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" httpsPort="9443"/>

<remoteFileAccess>
<writeDir>${server.config.dir}</writeDir>
<writeDir>${server.output.dir}</writeDir>
<writeDir>${wlp.user.dir}</writeDir>
</remoteFileAccess>
</server>
```

## Procedure

1. In the workbench, open the Servers view by clicking the **Servers** tab.




**Tip:** If the Servers view is not visible, navigate to **Window > Show view > Other...** and type **Server** in the filter text. Then, select **Servers**.

2. Right-click within the **Servers** view and select **New > Server**.
3. Under the server type list, expand **IBM** and select the **WebSphere Application Server Liberty** server type.
4. Enter the host name of the remote server.
5. Click **Next**. The Liberty Runtime Environment page is displayed.
- 6.





**Tip:** If you already have a Liberty run time that is installed, you will go directly to the **New Remote Liberty Server** page, skip to Step 7.

Select an installation, install from an archive file, or (for the no-charge developer edition) download and install Liberty.

If you previously installed Liberty, complete the following steps:

- a.  **8.5.5.4** Select **Choose an existing installation**.
- b.  **8.5.5.4** In the **Path** field, type or browse for the directory where you installed the Liberty runtime environment.
- c.  **8.5.5.4** On the Liberty Runtime Environment page, click **Next**.  
The application-serving environment is selected, and now you can skip to Step 7.

If you want to install Liberty from an archive file that was previously downloaded, complete the following steps:

- a.  **8.5.5.4** Select **Install from an archive or repository**, and click **Next**.
- b.  **8.5.5.4** In the **Destination** field, type or browse for the directory where you want to install the Liberty runtime environment.  
If you type a path that does not exist, then a folder for that path is created automatically at the end of Step 12.
- c.  **8.5.5.4** Select **Install a new runtime environment from an archive**.
- d.  **8.5.5.4** In the **Path** field, either type or browse to the archive file on the local file system, and click **Next**.
- e. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, click **Add Archive**. In the **Add-on archive** field, type or browse for the add-on archive file on the local file system, and then click **OK**. If you have more add-on archive files to install from your local file system, repeat this step until you are done.

If you want to install add-on files from a custom repository, first add the repository by clicking **Configure Repositories**. Click **New...** to add the repository. The files are then added to the list of add-on files.






After you complete your selection in the Install Add-ons page, click **Next**.

- f. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

Now you can skip to Step 7.

If you want to download and install the no-charge developer edition of Liberty, complete the following steps:

- a.  **8.5.5.4** Select **Install from an archive or repository**, and click **Next**.
- b.  **8.5.5.4** In the **Destination** field, type or browse for the directory where you want to install the Liberty runtime environment.  
If you type a path that does not exist, then a folder for that path is created automatically at the end of Step 12.
- c.  **8.5.5.4** Select **Download and install a new runtime environment from ibm.com**, choose a runtime environment version, and then click **Next**.
- d. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, click **Add Archive**. In the **Add-on archive** field, type or browse for the add-on archive file on the local file system, and then click **OK**. If you have more add-on archive files to install from your local file system, repeat this step until you are done.

If you want to install add-on files from a custom repository, first add the repository by clicking **Configure Repositories**. Click **New...** to add the repository. The files are then added to the list of add-on files.

After you complete your selection in the Install Add-ons page, click **Next**.

- e. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

Now you can skip to Step 7.

7. Complete the user ID, password, and port information for the remote server and click **Verify**.

**Important:**

- The user ID and password must have the appropriate security credentials as defined by the quickStartSecurity configuration item or the user registry of the remote Liberty server.
- The port is the HTTPS port that is configured in the server.xml file.

After you click **Verify**, if you see the following message, a local server or remote server is already created with the same name.

The Liberty server already exists

You can verify this situation by expanding the following folders in the Enterprise Explorer view and seeing the listed servers in the following projects:

- WebSphere Application Server Liberty
- WebSphere Application Server Liberty (Remote)

The remote directory is shown on the same page if the connection is successful. To resolve this issue of two servers with the same name, you can rename the Remote Directory folder on the remote system.

8. In the New Remote Liberty Server page, click **Next**.

The Remote WebSphere Application Server Settings page is displayed.

9. If you do not want to enable remote start, stop, and restart, ensure that **Enable the server to start remotely** is cleared, click **Next**, and skip the next step.

10. Enable remote start, stop, and restart.

- a. Select **Enable the server to start remotely**.

- b. Select whether your remote server is installed on the Windows operating system or other operating systems.

- c. Enter the location of the runtime installation and server configuration.

For more information, see the “Directory locations and properties” on page 894 topic.

- d. Enter the remote server authentication information.

For authentication, complete one of the following options.

- To access the remote server with logon credentials, enter your user name and password.
- To access the remote server with Secure Sockets Layer (SSL), copy the private key file to the computer where the workbench is installed, and specify the key file location and user ID.

For more information about starting a remote server, see the Starting a remote WebSphere Application Server topic.

11. Optional: Add the projects of your application to the server. On the **Add and Remove** page, under the **Available** list, select the projects that you want to add to the server and click **Add**. The project appears in the **Configured** list.

12. Click **Finish**.

## What to do next

- Edit the server configuration. When you make edits, the remote servers synchronize configuration changes with the remote version of the file. If the configuration files are out of sync with the remote server, then the developer tools prompt you before they overwrite any remote files. For more information, see topic “Editing the Liberty configuration by using developer tools” in the “Administering” book.
- Start the server in “start” mode or in “debug” mode, stop the server, add or remove applications on the server, and many other tasks. You can perform these tasks by using the server menu (right-click on the server to open the pop-up menu) or by selecting the tray buttons in the Servers view.

**Tip:** In the Servers view, you must select the server entry to perform these tasks. Do not select the server configuration, such as the Server Configuration [server.xml] entry for performing these tasks.

---

## Creating a workbench Liberty server in a Docker container by using developer tools

8.5.5.9

You can use WebSphere Developer Tools to create a reference in the workbench to a Liberty server that is running in a Docker container. You can use this reference to handle your server requests from the workbench.

### Before you begin

- Install WebSphere Developer Tools. For more information about installing WebSphere Developer Tools, see Installing the Liberty developer tools and (optionally) Liberty.

- Start a Liberty server in a local Docker container and ensure that both the HTTP and HTTPS ports are mapped to the host. For more information about starting a Liberty server in a local Docker container, see WASdev GitHub.

## About this task

**Remember:** You can create a reference in the workbench to a Liberty server that is running in a Docker container.

As you complete the steps, you can install WebSphere Application Server Liberty and download add-ons in the Liberty Repository from the WASdev community download site. After you install Liberty, you can select your Docker container. Then, you can enter the Liberty server security credentials for your Liberty server to create a workbench reference to your Liberty server in your Docker container.

## Procedure

1. In the workbench, open the Servers view by clicking the **Servers** tab.

**Tip:** If the Servers view is not visible, select **Window > Show view > Other...** and type Server in the filter text. Then, select **Servers**.

2. Right-click the **Servers** view and select **New > Server**.
3. To select your server type, select **IBM > WebSphere Application Server Liberty**.
4. Enter the localhost value in the **Server's host name** field.
5. Click **Next**.
6. If the tools prompt you to install WebSphere Application Server Liberty, either because it is not installed or because you need to input the directory where it is installed, complete one of the following options. Otherwise, skip this step.

You need Liberty to be installed so that you can create a workbench reference to a Liberty server in a Docker container.

If you previously installed Liberty, complete the following steps to select the directory where you installed it:

- a. Select **Choose an existing installation**.
- b. In the **Path** field, type or browse for the directory where you installed the WebSphere Application Server Liberty.
- c. On the Liberty Runtime Environment page, click **Next**.

The application-serving environment is selected. Skip the rest of this step.

If you want to install Liberty from an archive file that was previously downloaded, complete the following steps:

- a. Select **Install from an archive or repository**, and click **Next**.
- b. In the **Destination** field, type or browse to the directory where you want to install the Liberty runtime environment.  
If you type a path that does not exist, then a folder for that path is created automatically at the end of the procedure when you click **Finish**.
- c. Select **Install a new runtime environment from an archive**.
- d. In the **Path** field, either type or browse to the archive file on the local file system, and click **Next**.
- e. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download an add-on archive file, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, complete the following steps:

- 1) Click **Add Archive**.
- 2) In the **Add-on archive** field, type or browse to the add-on archive file on the local file system.
- 3) Click **OK**.

If you have more add-on archive files to install from your local file system, repeat this procedure until you are done.

If you want to install add-on files from a custom repository, complete the following steps:

- 1) To add the repository, click **Configure Repositories**.
- 2) Click **New...**

The files are then added to the list of add-on files.

After you complete your selection in the Install Add-ons page, click **Next**.

- f. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

If you want to download and install the no-charge developer edition for Liberty, complete the following steps:

- a. Select **Install from an archive or repository**, and click **Next**.
- b. In the **Destination** field, type or browse to the directory where you want to install the Liberty runtime environment.

If you type a path that does not exist, then a folder for that path is created automatically at the end of the main procedure when you click **Finish**.

- c. Select **Download and install a new runtime environment from ibm.com**, choose a runtime environment version, and then click **Next**.
- d. In the Install Add-ons page, click **Install** or **Install Pending** to make your selection of add-on archive files that you want to install on the Liberty runtime environment.

You can install an add-on archive file from your local file system, download, or use a combination of both when you install multiple add-on archive files. If the workbench is connected to the internet, the Install Add-ons page is populated with add-on archive files available for download from the WAS dev community download site.

This download site includes add-ons that you can select from the Liberty Repository, such as runtime features, samples, or open source integration.

If you want to install add-on archive files from your local file system, click **Add Archive**. In the **Add-on archive** field, type or browse for the add-on archive file on the local file system, and then click **OK**. If you have more add-on archive files to install from your local file system, repeat this step until you are done.

If you want to install add-on files from a custom repository, first add the repository by clicking **Configure Repositories**. Click **New...** to add the repository. The files are then added to the list of add-on files.

After you complete your selection in the Install Add-ons page, click **Next**.

- e. In the License Acceptance page, if you accept the license terms, select **I accept the terms of all the license agreements** then click **Next**.

7. On the Liberty Server Type page, select **Server in a Docker container**.

8. Click **Next**.

The tools display the New Liberty Server in Docker Container page.

9. Select your Docker container from the **Container name** menu.

10. Enter the Liberty server security credentials.

Your values for the **User name** and **Password** fields must correspond to a user that is defined in a user registry that has an administrator role. If the tools cannot find a basic user registry that defines the specified user, the tools display the Security Verification window. If the tools display the Security Verification window, complete one of the following options:

- Select **Create** to add a basic user registry that defines a user with your values and an administrator role.
  - Select **Proceed** to proceed if the user is defined in another user registry type.
  - Select **Cancel** to make changes to the **User name** and **Password** that you entered.
11. Enter the value for the **Host mapped secured port** field that corresponds to the HTTPS port in the server configuration.
  12. Click **Finish**.

## Results

You created a workbench server reference to a server that is running in a Docker container.

- To see the server that you created and its server configuration, open the Servers view.
- To see the project, the server folder, and the server configuration files, open the Enterprise Explorer view.

## What to do next

- You can edit the server configuration. For more information, see “Editing the Liberty configuration by using developer tools” on page 938
- To start or stop a server, or add or remove applications on the server, right-click your server to use the server context menu or use the tray buttons in the Servers view. For more information, see “Starting and stopping a server by using developer tools” on page 940.

**Remember:** To access these options, select the **Server Entry** in the Servers view. Do not select a server configuration such as the **Server Configuration** or `server.xml` entry.

**Tip:** The application address that the server logs in the Console view is valid within the Docker container only. To load an application in a browser, use the host mapped IP address and host mapped HTTP or HTTPS port. If you use the **Run on Server** or **Debug on Server** options, the application is automatically loaded with the correct IP address and port.

To make it easier to determine the correct port, map the ports to specific ports on the host by using the `-p` option rather than the `-P` option when you create the Docker container.

- You can configure your Liberty server to authenticate users with Tivoli Directory Server.
  1. Right-click **Servers**.
  2. Select **Utilities > Add config snippets**.
  3. Select the snippets and accept the licenses agreements.

The tools download the configuration snippets that you selected and add them to the `server.xml` file.

## Directory locations and properties

In Liberty, many directories have properties that are associated with them. These properties can be used to specify file locations when you configure the server.

*Table 70. Runtime environment default directory structure.* Column 1 contains a file and directory tree. If a directory has a property that is associated with it, this is given in column 2. A description of each file or directory is given in Column 3.

Directory or file	Property	Description
wlp/ +- bin/  +- clients/ +- jython/ +- dev/ +- api/ +- ibm/ +- javadoc/ +- spec/  +- third-party/          +- spi/ +- ibm/ +- javadoc/ +- spec/  +- tools/ +- etc/  +- server.env  +- client.env  +- jvm.options +- lafiles/ +- lib/ +- templates/   8.5.5.6 +- client/	wlp.install.dir	Root of installation  Scripts for managing the installation. For example, server.  Liberty client and thin client libraries. For example restConnector.jar.  Jython-based scripts  Root for developer resources (APIs, SPIs, specifications, and tools)  Public APIs available for both compile and run time by default  APIs available in Liberty  Java document archives  Public specification APIs available for both compile and run time by default  Third-party APIs that are available at compile time by default and must be specified in the configuration using the apiTypeVisibility attribute of the classloader element for applications at run time.  Public SPIs available for both compile and run time by default  SPIs available in Liberty  Java document archives for SPI  Public specification SPIs available for both compile and run time by default  Ant plug-in for Liberty  User customized server variables that apply to all servers (optional)  Default server script environment variables (optional)  Default client script environment variables (optional)  Default JVM options (optional)  License information files  Platform runtime environment  Runtime customization templates and examples  8.5.5.6 Client template when creating a client

Table 70. Runtime environment default directory structure (continued). Column 1 contains a file and directory tree. If a directory has a property that is associated with it, this is given in column 2. A description of each file or directory is given in Column 3.

Directory or file	Property	Description
+ server/		Server template when creating a server
+ usr/	wlp.user.dir	User directory
+ extension/	usr.extension.dir	User-developed features
+ shared/		
+ apps/	shared.app.dir	Shared applications
+ config/	shared.config.dir	Shared configuration files
+ resources/	shared.resource.dir	Shared resource definitions: adapters, data sources
8.5.5.7 +- stackGroups/	shared.stackgroup.dir	Shared stack groups for remote deployment of packages and installables
+ servers/		Shared servers directory
+ server_name	server.config.dir	Server configuration directory. Use <code>\${server.config.dir}</code> to reference server-specific configuration (applications).
+ bootstrap.properties		Server bootstrap properties (optional)
+ jvm.options		Server JVM options, which replace the values in <code>wlp/etc/jvm.options</code> (optional)
+ server.env		Server script environment variables, which are merged with <code>wlp/etc/server.env</code> (optional)
+ server.xml		Server configuration overlays (required)
+ apps/		Server configuration for applications
+ dropins/		Server default application dropins folder (optional)
+ application_name		Application folder or archive (optional)
8.5.5.5 +- configDropins/		Server configuration dropins folder (optional)
8.5.5.5 +- defaults		Default server configuration dropins folder (optional)
8.5.5.5 +- overrides		Server configuration overrides dropins folder (optional)
+ server_name	server.output.dir	Server output directory. Use <code>\${server.output.dir}</code> to describe artifacts generated by the server (log files and workarea).
+ logs/		Server log files, including FFDC logs (directory is present after server is first run)
+ console.log		Basic server status and operations messages
+ trace_timestamp.log		Time-stamped trace messages, with the level of detail determined by the current tracing configuration

Table 70. Runtime environment default directory structure (continued). Column 1 contains a file and directory tree. If a directory has a property that is associated with it, this is given in column 2. A description of each file or directory is given in Column 3.

Directory or file	Property	Description
<pre> +- ffdc/     +- ffdc_timestamp/ +- workarea/                     </pre>		<p>First Failure Data Capture (FFDC) output directory</p> <p>First Failure Data Capture (FFDC) output that typically includes selective memory dumps of diagnostic data related to the failure of a requested operation</p> <p>Files created by the server as it operates (directory is present after server is first run)</p>
<pre> 8.5.5.6 +- clients/ +- client_name +- bootstrap.properties +- client.jvm.options  +- client.xml +- apps/ +- logs/  +- trace_timestamp.log  +- ffdc/     +- ffdc_timestamp/  +- workarea/                     </pre>		<p>Shared clients directory</p> <p>Client configuration directory.</p> <p>Client bootstrap properties (optional)</p> <p>Client JVM options, which replace the values in <code>wlp/etc/client.jvm.options</code> (optional)</p> <p>Client configuration overlays (required)</p> <p>Client configuration for applications</p> <p>Client log files, including FFDC logs (directory is present after client is first run)</p> <p>Time-stamped trace messages, with the level of detail determined by the current tracing configuration</p> <p>First Failure Data Capture (FFDC) output directory</p> <p>First Failure Data Capture (FFDC) output that typically includes selective memory dumps of diagnostic data related to the failure of a requested operation</p> <p>Files created by the client as it operates (directory is present after client is first run)</p>

You can use the properties that are associated with each directory, if any, to specify file locations when you configure the server.

**Tip:** To ensure configuration portability, use the most specific property that is appropriate, and do not rely on the relationship between resources. For example, in some configurations the installation location, `${wlp.install.dir}` might not be the parent of the customized instance `${wlp.user.dir}`.

## Programmatic access to location properties

Location properties can be bound into the JNDI namespace under names of your choice, using the `jndiEntry` configuration elements in the `server.xml` file, for example:

```
<jndiEntry jndiName="serverName" value="${wlp.server.name}"/>
```



Such entries are accessible by any code that runs in the server (applications, shared libraries or features) through a JNDI lookup:

```
Object serverName = new InitialContext().lookup("serverName");
```

For more information on how to use JNDI entries in configuration, see “Using JNDI binding for constants from the server configuration files” on page 1336.

Feature code can also use a system programming interface (SPI) provided by the kernel to resolve the values of these properties, for example:

```
ServiceReference <WsLocationAdmin>locationAdminRef = bundleContext.getServiceReference(WsLocationAdmin.class);
WsLocationAdmin locationAdmin = bundleContext.getService(locationAdminRef);
String serverName = locationAdmin.resolveString("${wlp.server.name}");
```

---

## Specifying Liberty bootstrap properties

Bootstrap properties initialize the runtime environment for a particular server. Generally, they are attributes that affect the configuration and initialization of the runtime core.

### About this task

Bootstrap properties are set in a text file named `bootstrap.properties`. This file is not required, so it does not exist unless you create it. You must create this file in the server directory, which also contains the configuration root file `server.xml`. By default, the server directory is `usr/servers/server_name`. You can change the server directory as described in “Customizing the Liberty environment” on page 947.

You can create a `bootstrap.properties` file by using the editor in WebSphere Application Server Developer Tools for Eclipse. From the **Servers** view, right-click on the server you want to configure, then select **New**, then **Server Environment File**, then `bootstrap.properties`, and the file is created from a template and opened in an editor. Along with the `server.xml` and `server.env` files, the `bootstrap.properties` file appears in the **Servers** view under the server that it is associated with and can be edited by double-clicking it.

You can edit the `bootstrap.properties` file by using a text editor or the editor in the WebSphere Application Server Developer Tools for Eclipse. See “Editing the Liberty configuration by using developer tools” on page 938.

If you update the `bootstrap.properties` file, you must restart the server for the changes to take effect.

The `bootstrap.properties` file contains two types of properties:

- A small, predefined set of initialization properties.
- Any custom properties that you choose to define. You can then use these custom properties as variables in other configuration files such as `server.xml` and included files.

If you update the `bootstrap.properties` file, you must restart the server for the changes to take effect.

### Procedure

- Use predefined properties to configure trace and logging.

For example:

- To change the name of your trace file, specify the property `com.ibm.ws.logging.trace.file.name` with a file name of your choice, as follows:

```
com.ibm.ws.logging.trace.file.name = trace.log
```

- To enable binary logging, specify the `websphere.log.provider` property as follows:

```
websphere.log.provider = binaryLogging-1.0
```

For more information, see “Configuring binary logging in Liberty” on page 629

- Use predefined properties for OSGi framework diagnostics. For example, set the port for the OSGi console as follows:

```
osgi.console = 5678
```

For more information, see “Using an OSGi console” on page 967

- Use predefined properties for OSGi framework extensions. Specify the **org.osgi.framework.bootdelegation** if this property is required by external monitoring tools. The value is a comma-delimited list of packages.
- Use predefined properties for configuration password encryption. For more information, see “The limits to protection through password encryption” on page 616.
- Use custom properties to define the default ports for web applications.

You can share `server.xml` and use XML configuration files across various development environments that allow machine- or environment-specific customization. For example:

1. Specify the properties **default.http.port** and **default.https.port** in the `bootstrap.properties` file:

```
default.http.port = 9081
default.https.port = 9444
```

**Note:** If you do not specify the properties, the default HTTP port is 9080 and HTTPS ports is 9443. To override the default HTTP endpoint definition, set the `id` attribute of the `httpEndpoint` element to `defaultHttpEndpoint` in the server configuration.

2. Use the following properties in the `server.xml` configuration file:

```
<httpEndpoint id="defaultHttpEndpoint"
host="*"
httpPort="${default.http.port}"
httpsPort="${default.https.port}" />
```

**Note:** `host="*" means “listen on all adapters”. By default, the server is listening only on address 127.0.0.1/localhost. You can also use the host property to specify a single IP address, and then the system listens only on the specified adapter.`

- **8.5.5.2** Use custom properties to set the command port.

Set the command port to enable the server script to communicate with the running Liberty server and request certain operations, such as stopping the Liberty server or issuing a Java dump.

#### Distributed operating systems

IBM i

By default, the Liberty server acquires an ephemeral port to be used by the command listener. You can override the default behavior of the Liberty server by using the **command.port** property.

#### Valid values

- 1 Command port is disabled.
- 0 Ephemeral port is chosen at run time.
- 1-65535 User-specified port.

#### Default value

- 0 **Distributed operating systems** IBM i Ephemeral port is chosen at run time.

**Note:** **Distributed operating systems** IBM i You are discouraged from disabling the command port. If you disable the command port, you cannot use the server script to request some operations, for example, stopping the Liberty server or issuing a Java dump.

- Use custom properties to configure server start wait time.

You can increase the server start wait time beyond the product default setting by adding the `server.start.wait.time` property to the `bootstrap.properties` file. The `server.start.wait.time` is specified in seconds.

1. Specify the `server.start.wait.time` property in the `bootstrap.properties` file. The following example sets the server start time to 25 seconds.

```
server.start.wait.time = 25
```

This setting means that as the server starts, the reporting mechanism for the server attempts to report on the completed stages of the start. If the reporting mechanism for the server cannot perform its function within 25 seconds, an error 22 occurs.

If you do not add the `server.start.wait.time` property to the `bootstrap.properties` file, the default server start wait time is internally set to 30 seconds.

- To apply the changes, restart the server.

---

## Setting the default host name of a Liberty server

You can add the `defaultHostName` variable to the `server.xml` file to set the default host name by which a Liberty server is identified.

### About this task

This variable is available for use by service configurations. Setting this value is particularly important for multihomed systems (with multiple NICs and multiple mapped host names for example).

#### Notes:

- This variable is currently used by the `<httpEndpoint>` host attribute and the `<hostAuthInfo>` `rpcHost` attribute
- The default host name should be the fully qualified host name of the system.
- The value should not contain any wildcards (\* for example).
- The variable default is `localhost`.

### Procedure

To set the default host name by which a Liberty server is identified, add the `defaultHostName` variable to the `server.xml` file.

For example:

```
<variable name="defaultHostName" value="localhost" />
```

---

## Default port numbers

Some parts of Liberty use default TCP/IP port numbers. You can override the default port numbers by specifying a different port number in your server configuration.

### Runtime environment port numbers

For the command port, the Liberty server acquires an ephemeral port to be used by the command listener. You can configure this port in the `bootstrap.properties` file. For more information, see “Specifying Liberty bootstrap properties” on page 897.

### Feature port numbers

The following table lists the default port numbers of Liberty features and an example of how you can override the default port in your server configuration.

Table 71. Default port numbers of Liberty features

Feature	Default port and configuration example
8.5.5.7 rtcomm-1.0	<ul style="list-style-type: none"> <li>• MQTT over TCP port: 1883</li> <li>• MQTT over SSL port: 8883</li> </ul> <pre>&lt;mqttTcpEndpoint id="defaultMqttEndpoint"   mqttTcpPort="1885"   mqttSslPort="8885"/&gt;</pre>
servlet-3.0	<ul style="list-style-type: none"> <li>• HTTP port: 9080</li> <li>• HTTPS port: 9443</li> </ul> <pre>&lt;httpEndpoint id="defaultHttpEndpoint"   httpPort="9082"   httpsPort="9445" /&gt;</pre>
8.5.5.7 sipServlet-1.1	<ul style="list-style-type: none"> <li>• TCP port: 5060</li> <li>• TLS port: 5061</li> <li>• UDP port: 5060</li> </ul> <pre>&lt;sipEndpoint id="defaultSipEndpoint"   sipTCPPort="5062"   sipUDPPort="5062"   sipTLSPort="5063" /&gt;</pre>

## Using virtual hosts

8.5.5.7

You can use virtual hosts if you want isolation between applications and the endpoints that serve them.

A single application server is often responding to requests from multiple different host and port configurations. This occurs for a combination of reasons, such as it is running on a machine with multiple network interfaces with different names or it is routed to from an http server, proxy, or load balancer. In these cases, you might want to control which application can be contacted from a specific host. Virtual hosts provide this capability. It matches the requested host name and port number (as determined from the HTTP Host header) against the configured list of host aliases.

In WebSphere Application Server Liberty, the default configuration is sufficient. The default virtual host (default\_host) matches requests from any incoming host and port combination, and forwards them on to the default application container.

The following list illustrates the key configuration elements when you are configuring virtual hosts.

- The virtualHost configuration element ID value.
- The hostAlias subelement configuration.
- The allowFromEndpoint subelement configuration (if used).
- The virtual host configuration in the `ibm-web-bnd.xml` or `ibm-web-bnd.xmi` file of the WAR.
- The host attribute value of the `httpEndpoint`.
- The ID attribute value of the `httpEndpoint`.

## Isolating two applications from each other

The following example illustrates one of the more common usages of virtual hosting to give an understanding of some of the configuration that is required. This example shows how to configure two applications that run on differing ports. Further in this example illustrates that one application is only available on the localhost interface.

```

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" />
<httpEndpoint id="alternateEndpoint" host="*" httpPort="9081" />

<virtualHost id="application-1">
 <hostAlias>your_host_name:9080</hostAlias>
</virtualHost>

<virtualHost id="application-2">
 <hostAlias>localhost:9081</hostAlias>
</virtualHost>

<enterpriseApplication location="myApp.ear" name="App1"/>
<webApplication location="myApp2.war" name="App2" />

```

The defaultHttpEndpoint exposes all interfaces on port 9080, and the alternateEndpoint exposes all interfaces on port 9081.

If App1 has a WAR file with an ibm-web-bnd.xml file that specifies <virtual-host name="application-1"/>, then this application can be accessed only at your\_host\_name:9080/app1\_context\_root.

If App2 (which is a WAR) has an ibm-web-bnd.xml file that specifies <virtual-host name="application-2"/>, then this application can be accessed only at localhost:9081/app2\_context\_root.

If a third application was deployed which specified no specific virtual host, in this configuration, that application would be accessible only if it were a proxied request that contained HOST header that specifies a different port. For example, if the request was made to a proxy on port 80, that port is not listed in any of the hostAlias specifications, and so the request would be routed to the default\_host virtual host.

## Isolating applications based on the requested host or port

The default virtual host in Liberty is also used for JMX communications. If you wanted to isolate JMX communications from application traffic, you would need to complete the following steps.

1. Decide on your virtual host name, and update your application to reference the new (non-default) host. Add a virtual-host element to the ibm-web-bnd.xml or ibm-web-bnd.xmi file of the WAR.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-bnd
 xmlns="http://websphere.ibm.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://websphere.ibm.com/xmk/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-bnd_1_0_0
 version="1.0" />

 <virtual-host name="proxiedRequests" />

</web-bnd>

```

2. Add a virtualHost element to your server.xml file. The name must match what is specified in the application, and must define hostAliases that are routed to the new virtual host.

**Note:** The host name and the port that is being matched is the one that is originally requested by the user, which might or might not match the host and port that Liberty is using. The following example illustrates a virtual host element added to your server.xml file.

```

<virtualHost id="proxiedRequests">
 <hostAlias>external.host.name:80</hostAlias>
 <hostAlias>external.host.name:443</hostAlias>
</virtualHost>

```

If requests are coming from a proxy, this configuration alone routes any request that is made to the proxy's host and port to the "proxiedRequests" virtual host.

## Restricting access based on originating endpoint

If you want to restrict access to the default/system applications that are using the defaultHttpEndpoint, there are more steps to take.

1. Define another httpEndpoint. The following example illustrates another httpEndpoint.

```
<httpEndpoint id="localhostOnly" host="localhost" httpPort="9081" httpsPort="9444"/>
```

This http endpoint specifies that host="localhost," meaning that ports 9081 and 9444 are exposed only on the localhost interface.

2. Update virtualHost definitions to specify the allowFromEndpointRef attribute. When this attribute is specified, a virtualHost accepts requests only from the specified endpoint. For example:

```
<virtualHost id="default_host" allowFromEndpointRef="localhostOnly">
 <hostAlias>*:9081</hostAlias>
 <hostAlias>*:9444</hostAlias>
</virtualHost>
```

```
</virtualHost id="proxiedRequests">
 <hostAlias>*:9080</hostAlias>
 <hostAlias>*:9443</hostAlias>
 <hostAlias>external.host.name:80</hostAlias>
 <hostAlias>external.host.name:443</hostAlias>
</virtualHost>
```

With this configuration:

- The default\_host virtual host now accepts requests that are directed only at localhost:9081 and localhost:9444 that also originate from the localhostOnly endpoint. Any other request to ports 9081 and 9444 are refused. For example, a request from the defaultHttpEndpoint with Host headers that reference localhost:9081 is refused.
- The proxiedRequests virtual host now accepts any request that is issued to port 9080, or 9443 (which are the default ports that are used by the defaultHttpEndpoint), in addition to those that have a Host header that references the external host name from the proxy and port 80 or 443.

## Virtual hosts

8.5.5.7

A virtual host is a configuration entity that enables a single host machine to resemble multiple host machines.

A virtual host maintains a list of Multipurpose Internet Mail Extensions (MIME) types that it processes. You can associate a virtual host to one or more Web modules, but you can associate each web module with only one virtual host. Resources that are associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share a physical machine.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP hostname and port number that is used to request the servlet, for example yourHostName:80. When no port number is specified, 80 is assumed.

The virtual host configuration uses wildcard entries with the ports for its virtual host entries.

- The default alias is \*:80 using an external port that is not secure.
- Aliases of the form \*:9080 use the internal port that is not secure.
- Aliases of the form \*:9443 use the secure internal port.
- Aliases of the form \*:443 use the secure external port.

A client request for a servlet, JavaServer Pages file, or related resource, contains a DNS alias and a Uniform Resource Indicator (URI) that is unique to that resource. When a client request for a servlet, JavaServer Pages file, or related resource is received, the DNS alias is compared to the list of all known virtual host groups to locate the correct virtual host. The URI is compared to the list of all known URI

groups to locate the correct URI group. If the virtual host group and URI group are found, the request is sent to the corresponding server group for processing and a response is returned to browser. If a matching virtual host group or URI group is not found, an error is returned to the browser.

A virtual host is not associated with a particular node or machine. It is a configuration, rather than a live object, which is why you can create it, but cannot start or stop it. A default virtual host, named `default_host`, is automatically configured the first time that you start an application server. Unless you specifically want to isolate resources from one another on the same node, or physical machine, you probably do not need any virtual hosts in addition to the default host.

The DNS aliases for the default virtual host are configured as `*:80` and `*:9080`, where port 80 is the HTTP server port and port 9080 is the port for the default server's HTTP transport. The default virtual host includes common aliases, such as the machine's IP address, short host name, and fully qualified host name. One of these aliases comprises the first part of the path for accessing a resource such as a servlet. For example, the alias `localhost:80` is used in the request `http://localhost:80/myServlet`.

When you request a resource, the product tries to map the request to an alias of a defined virtual host. The `http://host:port/` portion of the virtual host is not case sensitive, but the URL that follows is case sensitive. The match for the URL must be alphanumerically exact. Different port numbers are treated as different aliases.

For example, the request `http://www.myhost.com/myservlet` maps successfully to `http://WWW.MYHOST.COM/myservlet` but not to `http://WWW.MYHOST.COM/MYSERVLET` or `Www.Myhost.Com/MyServlet`. In the latter two cases, these mappings fail because of case sensitivity. The request `http://www.myhost.com/myservlet` does not map successfully to `http://myhost/myservlet` or to `http://myhost:9876/myservlet`. These mappings fail because they are not alphanumerically correct.

You can use wildcard entries for aliases by port and specify that all valid host name and address combinations on a particular port map to a particular virtual host.

If you request a resource by using an alias that cannot be mapped to an alias of a defined virtual host, you receive a 404 error in the browser that you used to issue the request. A message states that the virtual host could not be found.

Two sets of associations occur for virtual hosts. Application deployment associates an application with a virtual host. Virtual host definitions associate the network address of the machine and the HTTP transport or web server port assignment of the application server with the virtual host. Looking at the flow from the web client request for the snoop servlet, for example, the following actions occur:

1. The web client asks for the snoop servlet: at web address `http://www.some_host.some_company.com:9080/snoop`
2. The `some_host` machine has the 9080 port that is assigned to the stand-alone application server, `server1`.
3. `Server1` looks at the virtual host assignments to determine the virtual host that is assigned to the alias `some_host.some_company.com:9080`.
4. The application server finds that no explicit alias for that DNS string exists. However, a wildcard assignment for host name `*` at port 9080 does exist, so this wildcard is a match. The virtual host that defines the match is `default_host`.
5. The application server looks at the applications that are deployed on the `default_host` and finds the snoop servlet.
6. The application server serves the application to the web client and the requester is able to use the snoop servlet.

Table 72. Aliases for a virtual host

Virtual host	Alias	Port number
default_host	*	9080
default_host	localhost	9080
default_host	my_machine	9080
default_host	my_machine.my_company.com	9080
default_host	localhost	80

You can have any number of aliases for a virtual host. You can even have overlapping aliases, such as:

The Application Server looks for a match by using the explicit address that is specified on the web client address. However, it might resolve the match to any other alias that matches the pattern before it matches the explicit address. Defining an alias first in the list of aliases does not guarantee the search order whenever the product is looking for a matching alias.

Virtual hosts with overlapping aliases. Assume that you define overlapping aliases for both virtual hosts because you accidentally defined port 9080 for the admin\_host instead of port 9060:

A problem can occur if you use the same alias for two different virtual hosts. For example, assume that you installed the default application and the snoop servlet on the default\_host. You also have another virtual host that is called the admin\_host. However, you have not installed the default application or the snoop servlet on the admin\_host.

Table 73. Virtual hosts with overlapping aliases.

Virtual host	Alias	Port number
default_host	*	9080
default_host	localhost	9080
admin_host	*	9060
admin_host	my_machine.com	9080

Assume that a web client request comes in for `http://my_machine.com:9080/snoop`.

If the application server matches the request against `*:9080`, the application is served from the default\_host. If the application server matches the request to `my_machine.com:9080`, the application cannot be found. A 404 error occurs in the browser that issues the request. A message states that the virtual host could not be found.

This problem is the result of not finding the requested application in the first virtual host that has a matching alias. The correct way to code aliases is for the alias name on an incoming request to match only one virtual host in all of your virtual host definitions. If the URL can match more than one virtual host, you see the problem that is described.

---

## Preparing and running an application client

8.5.5.6

Learn how to prepare your server and client to run an application client from Liberty application client container.



## About this task

Running an application client successfully requires updates to both the `server.xml` and `client.xml` files.

## Procedure

Prepare your server, as follows:

1. Package a client module (.jar) and other modules, such as an EJB module (.jar), in an application EAR file.
2. Place the EAR file in the apps directory; for example, `wlp/usr/servers/your_server/apps`.
3. Update the `server.xml` configuration file by adding the `appClientSupport-1.0` feature, along with other necessary features.

**Important:** This step is not required if your application client is a stand-alone application.

4. Update the `server.xml` configuration file by configuring `<application/>` with your application information; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
 <!-- Enable features -->
 <featureManager>
 <feature>javaee-7.0</feature>
 </featureManager>
 <application id="techsample" name="techSample" type="ear" location="TechnologySamples.ear"/>
</server>
```

Prepare your client, as follows:

5. Place the EAR file in the apps directory; for example, `wlp/usr/clients/your_client/apps`.
6. Update the `client.xml` configuration file by configuring `<application/>` with your application information; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<client description="new client">
 <!-- Enable features -->
 <featureManager>
 <feature>javaeeClient-7.0</feature>
 </featureManager>
 <application id="techsample" name="techSample" type="ear" location="TechnologySamples.ear"/>
</client>
```

7. Optional: Add the `appClientSecurity-1.0` feature to the `client.xml` file. Read about “Creating a Liberty application client manually” on page 906.
8. Start the server.
9. Run the client by entering `client run your_client`. If your client application uses command-line arguments, use the following format:  
`client run {your_client} -- arg1 arg2 ... argn`

There are additional steps to take if your server and client are running on different computers. By default, the server and client are using `localhost:2809`. You must configure IIOP to establish a connection between the server and client, as follows:

10. Stop the server.
11. Update the `server.xml` file with the IIOP configuration; for example:  
`<iiopEndpoint id="defaultIiopEndpoint" host="user.host.ibm.com" iiopPort="2814" />`
12. Update the `client.xml` file with the IIOP configuration; for example:  
`<orb id="defaultOrb" nameService="corbaname::user.host.ibm.com:2814" />`
13. Start the server.

# Creating a Liberty application client manually



8.5.5.6

You can create a Liberty application client from the command line.

## Before you begin

You enable the Java EE Application Client 7.0 feature in the `client.xml` file only.

## Procedure

1. Open a command line, then change directory to the `wlp/bin` directory. In the following examples, `path_to_liberty` specifies the location where you installed Liberty on your operating system.

**Windows** Example on Windows systems: `C:\Users\mo> cd path_to_liberty\wlp\bin`

**Linux** Example on Linux: `mo@machine01:~> cd path_to_liberty/wlp/bin`

2. Run the following command to create a client, where `client_name` is the name that you want to give your client. If you do not specify a client name, `defaultClient` is used.

**Windows** Example on Windows systems: `C:\wlp\bin> client create client_name`

**Linux** Example on Linux: `mo@machine01:~> client create client_name`

`client create client_name`

If the client is created successfully, you receive the following message:

Client `client_name` created.

You can find the `client.xml` file in the `wlp/usr/clients/client_name` directory. The file contains the `javaeeClient-7.0` feature.

**Attention:** If a default client exists, you get an error. If a default client does not exist, `defaultClient` is created.

3. Run your client application by preparing an application (`.ear`) file with a client module (`.jar`) in it. Specify a main class in the `MANIFEST.MF` of the client module, for example:

```
Manifest-Version: 1.0
Main-Class: com.ibm.ws.addressbook.ContactServiceClient_XMLInject
```

4. Place the EAR file under the `wlp/usr/clients/client_name/apps` directory.
5. Update the `client.xml` file to configure your application, for example:

```
<client>
 <featureManager>
 <feature>javaeeClient-7.0</feature>
 </featureManager>
 <application id="CLIENT_APP" name="CLIENT_APP" type="ear" location="clientApp.ear"/>
</client>
```


If the specified client already exists, no client is created and you receive an exception message:

```
CWWKE0005E: The runtime environment could not be launched.
CWWKE0904E: It was not possible to create the client called client_name because
the client directory C:\wlp\usr\clients\client_name already exists.
```

## What to do next

You can enable security (SSL, CSiv2, JAAS) for your application client by adding the `appSecurityClient-1.0` feature to your `client.xml` file:

```
<featureManager>
 <feature>javaeeClient-7.0</feature>
 <feature>appSecurityClient-1.0</feature>
</featureManager>
```

For more information about configuring security on the application client, see  **8.5.5.6** “Configuring security for the Liberty application client container and its applications” on page 1287.

## Creating a Liberty application client with multiple client modules

**8.5.5.6**

You can create a Liberty application client with multiple client modules in the same EAR file.

### About this task

You can specify more than one client application (packaged in a client module) in your application EAR file. If you would like to package multiple client applications in the same EAR file, you must use the `defaultClientModule` attribute in `<enterpriseApplication/>`.

### Procedure

Specify the client that you want to run by updating the `defaultClientModule` attribute in both the `client.xml` and `server.xml` files.

```
<client>
 <featureManager>
 <feature>javaeeClient-7.0</feature>
 </featureManager>
 <enterpriseApplication id="MultipleAppClientModules" name="MultipleAppClientModules" type="ear"
 defaultClientModule="HelloAppClient.jar" location="MultipleAppClientModules.ear"/>
</client>
```

**Important:** You can run one application at a time.

---

## Setting up the server-management environment for Liberty by using collectives

To set up the server-management environment for Liberty by using collectives, define the appropriate features in the `server.xml` file and run the corresponding collective command-line tasks to establish the administrative domain security configuration.

### About this task

You can use collectives to manage multiple servers from a single management domain. For high availability, you can configure collective replica sets, clusters, or scaling. For general information about collectives, see “Collective architecture” on page 908.

Liberty provides multiple-server management in the following features:

- **collectiveController-1.0**

The `collectiveController-1.0` feature enables controller functionality for a management collective and includes collective- and cluster-management MBeans that are accessible using the REST JMX connector that is provided by the `restConnector-1.0` feature. The collective controller acts as a storage and collaboration mechanism to which collective members can connect. The administrative domain security configuration for the `collectiveController-1.0` feature is established using the **collective** command-line **create** and **replicate** tasks.

**Distributed operating systems**

**IBM i**

The `collectiveController-1.0` feature and its capabilities are available only in multiple-server products such as WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in single-server products such as WebSphere Application Server Liberty, WebSphere Application Server

Liberty - Express, or WebSphere Application Server Liberty Core. If you have a multiple-server product installation, you can use its `collectiveController-1.0` feature to work with collective members from single-server products.

- **collectiveMember-1.0**

The `collectiveMember-1.0` feature enables a server to be a member of a management collective and be managed by the collective controller. The administrative domain security configuration for the `collectiveMember-1.0` feature is established using the **collective** command-line **join** task.

**Tip:** All servers enabled with the `collectiveController-1.0` feature are managed; therefore, you do not need to specify `collectiveMember-1.0` if the server already has the `collectiveController-1.0` feature enabled.

## Procedure

- Configure a server to act as collective controller.
  - Create a collective controller server using commands and then configure the controller `server.xml` file. See step 1 of “Configuring a Liberty collective” on page 912.
  - Create a collective controller using developer tools. See step 1 of “Configuring a Liberty collective using the developer tools” on page 918.
- Join a server to a collective as a member.
  - Join a server to a collective using commands and then configure the member `server.xml` file. See step 2 of “Configuring a Liberty collective” on page 912.
  - Join a server to a collective using developer tools. See step 2 of “Configuring a Liberty collective using the developer tools” on page 918.
- Change the collective configuration as needed for your environment.
  - Register host computers with a collective.
  - Override host information by revising parameters in the `hostAuthInfo` element of the member `server.xml` file.
  - Set up Remote Execution and Access (RXA) to enable collective controllers to start and stop servers.
  - Set the `JAVA_HOME` variable for Liberty collective members.

## Collective architecture

The set of Liberty servers in a single management domain is called a *collective*. A collective consists of at least one server with the `collectiveController-1.0` feature enabled that is called a *collective controller*. Optionally, a collective can have many servers with the `collectiveMember-1.0` feature enabled that are called *collective members* and a collective can be configured to have many collective controllers.

**Note:** **Distributed operating systems** **IBM i** The `collectiveController-1.0` feature and its capabilities are available only in multiple-server products such as WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in single-server products such as WebSphere Application Server Liberty or WebSphere Application Server Liberty Core. If you have a multiple-server product installation, you can use its `collectiveController-1.0` feature to work with collective members from single-server products.

The collective controller provides for a centralized administrative control point to perform operations such as MBean routing, file transfer, and cluster management. A core role of collective controllers is to receive information, such as MBean attributes and operational state, from the members within the collective so that the data can be retrieved readily without having to invoke an operation on each individual member.

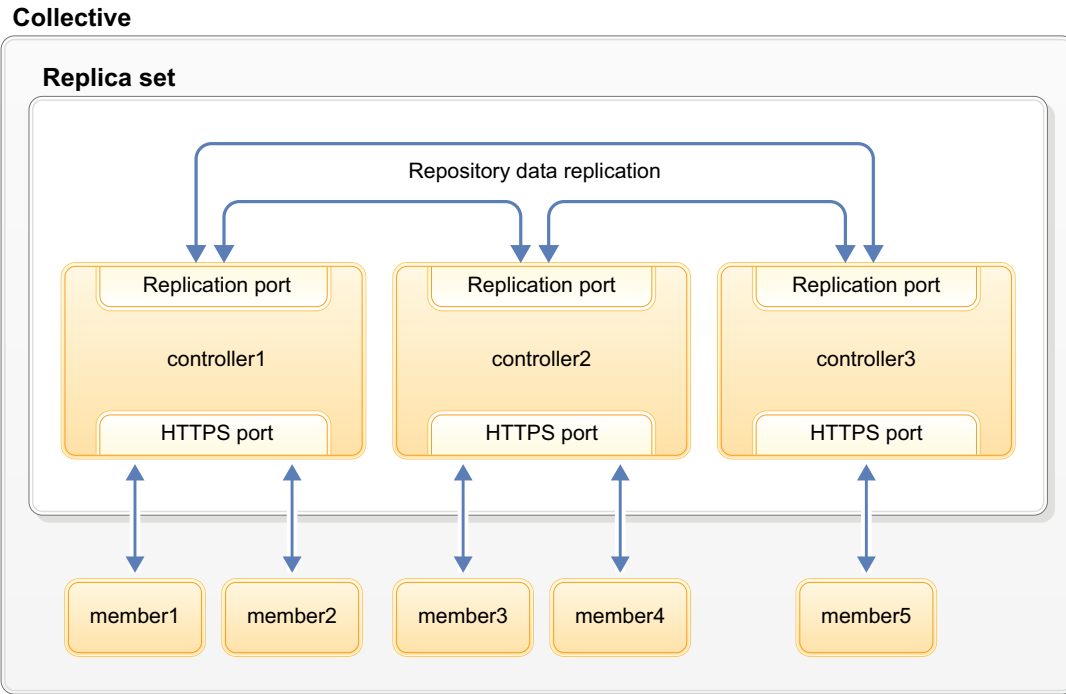


Figure 15. Liberty collective architecture

A set of collective controllers is called a *replica set*. There is only one replica set per collective, and all controllers must be part of the replica set. When there is more than one collective controller, each collective controller replicates its data to the other collective controllers in the replica set to allow for high availability and data protection. The replica set is logically present even when only one controller is in use. When changing your configuration to multiple replicas in a set, include at least three replicas in the set. The controllers in the replica set communicate with each other using a collaboration scheme to ensure that data is replicated across the set of controllers no matter which controller in the set receives an operation to store data. Each controller has a dedicated port for use by the replication protocol. Communication between the controllers in the replica set is always authenticated and protected with SSL.

A collective member can be configured with multiple collective controller endpoints. A collective member only communicates with one collective controller at a time; however, a configuration with more than one collective controller endpoint provides failover and workload balancing. Member-to-controller communication is always in the form of MBean operations that are performed over the IBM JMX Rest Connector. Communication between controllers and members is always authenticated and protected with SSL.

See “Setting up the server-management environment for Liberty by using collectives” on page 907 for more information.

#### Administrative domain security configuration:

The administrative domain security configuration is made up of two parts:

- User domain  
This domain relies on Java role-based security that defines the Administrator role. This can be mapped to users within the configured user registry.
- Server domain  
This domain relies on SSL certificate-based authentication.

For more on collective security, see “Collective security” on page 911.

## Configured and standby replicas

Replicas that have been added to a configured replica set are running (*active replicas*) or stopped (*inactive replicas*). A replica that is started and that has never been added to a configured replica set, or was removed from a configured replica set, is called a *standby replica*.

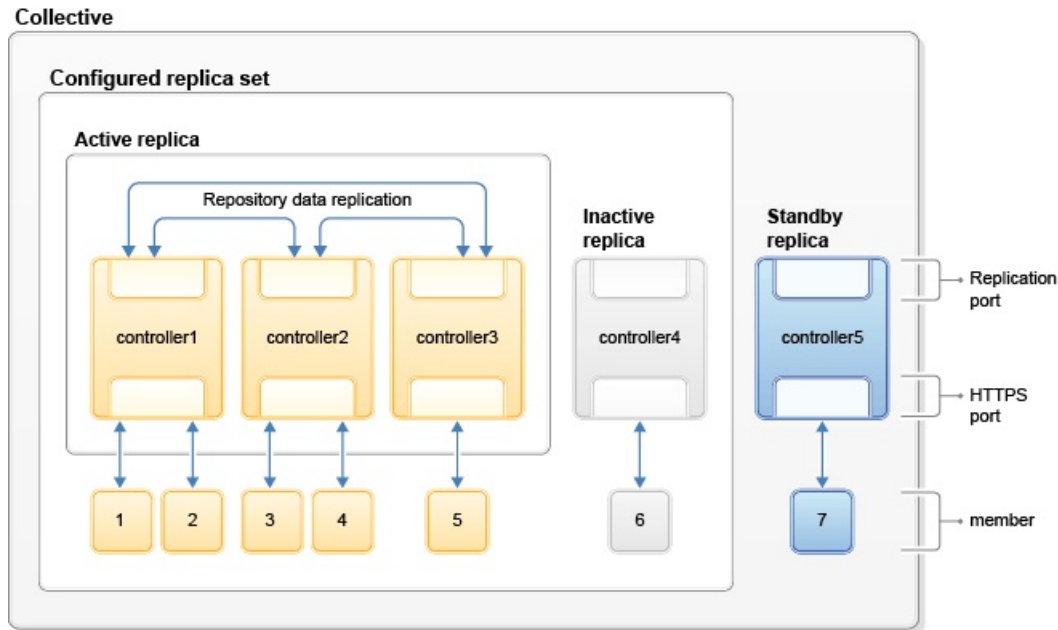


Figure 16. Configured and standby replicas in a collective controller

## Summary of collective architecture terms

### collective

The set of Liberty servers in a single management domain.

### collective controller

A server that has the `collectiveController-1.0` feature enabled.

### collective member

A server that has the `collectiveMember-1.0` feature enabled.

### replica set

A set of collective controllers. For optimal functionality and high availability, a replica set must have at least three controllers.

### replica port

A dedicated port on a controller that is used by the replication protocol.

### configured replica set

The union of the active replicas and inactive replicas.

### active replicas

The started replicas that have been added to the configured replica set.

### inactive replicas

The stopped replicas that have been added to the configured replica set.

### standby replica

The started replicas that have not been added to the configured replica set or that were removed from the configured replica set.

## Collective security

You can use the principles of collective security in Liberty to address data in motion and data at rest.

The two principal areas of collective security are:

- Administrative domain security configuration  
Addresses data in motion, authentication and authorization
- Collective repository data security  
Addresses data at rest, authentication and authorization

### Administrative domain security configuration

The administrative domain security configuration for collectives is comprised of two parts:

- User domain  
This domain relies on Java role-based security that defines the Administrator role. This can be mapped to users within the configured user registry.
- Server domain  
This domain relies on SSL certificate-based authentication.

In order for users to access the collective controller's MBeans, they must be in the Administrator role. All administrative actions through the collective require that the user be granted the Administrator role. See “Configuring secure JMX connection to Liberty” on page 1022 for complete details.

Server-to-server communication falls within the server domain and no user identities or passwords are used to communicate between members of a collective. Each member of the collective has a unique identity within the collective that is comprised of its host name, user directory, and server name. Each member within the collective defines its server domain configuration, which consists of the `serverIdentity.jks` and `collectiveTrust.jks` files. These files contain the SSL certificates that are necessary to establish secure communications within the collective. The HTTPS key configuration must have specific trust settings, which are established by default.

The server domain SSL configuration can be customized by adding additional trusted certificate entries to the `collectiveTrust.jks` keystore. All trust is copied when a controller is replicated; therefore, SSL customization should be applied to the initial controller. Adding trust to the `collectiveTrust.jks` keystore is only necessary if the default HTTPS certificates are not used. If the HTTPS SSL configuration is modified, the following certificate rules apply:

- HTTPS trust must be established by all controllers and members within the collective. If the HTTPS SSL certificates are modified, the following root signers from the collective controller must be added to the HTTPS SSL truststore:
  - The `controllerRoot` signer from the `rootKeys.jks` keystore must be added to all collective members HTTPS SSL truststore.
  - The `controllerRoot` signer and the `memberRoot` signer from the `rootKeys.jks` keystore must be added to all collective controllers' HTTPS SSL truststore.
- Each member can make an outbound connection to a collective controller. The collective controller's `collectiveTrust.jks` keystore must contain a certificate chain that trusts the HTTPS SSL certificate for each member. It is highly recommended that all HTTPS certificates be signed by a root signer, which then can be added to the `collectiveTrust.jks` keystore. Using individual SSL certificates that do not have a common root signer is sufficient to establish trust but will not scale.
- Each controller can make an outbound connection to a collective member. The collective member's `collectiveTrust.jks` keystore must contain a certificate chain that trusts the HTTPS SSL certificate for each controller. It is highly recommended that all HTTPS certificates be

signed by a root signer, which then can be added to the `collectiveTrust.jks` keystore. Using individual SSL certificates that do not have a common root signer is sufficient to establish trust but will not scale.

Server-to-server communication requires that SSL authentication be supported. If the HTTPS SSL configuration is customized, the SSL configuration must specify `clientAuthenticationSupported="true"`. For example:

```
<!-- clientAuthenticationSupported set to enable bidirectional trust -->
 <ssl id="defaultSSLConfig"
 keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore"
 clientAuthenticationSupported="true" />
```

Setting `clientAuthentication="true"` on the collective controller is inadvisable and prevents some common and expected behaviors. For example, this setting prevents authentication with user names and passwords in Admin Center and collective command line utilities.

Setting `clientAuthentication="true"` on a collective member might be desirable to prevent user name and password logins. This setting does not break collective operations as all operations originating from the controller are authenticated using the certificate.

Members can be prevented from publishing information to the collective controller by using the `CollectiveRegistration` MBean. The `disavow` and `avow` methods prevent authentication and enable authentication.

### Collective repository data security

The collective repository data security policy covers the policy for data at rest - specifically, the policy of accessing the contents of the collective repository.

The current security policy for collective data is as follows:

- The system reserves three node names: `sys.host.auth.info`, `sys.jmx.auth.info`, and `sys.nologin`. These nodes are under a host or server's repository namespace. User-created nodes should avoid using the `sys.` prefix.
- The `sys.host.auth.info` and `sys.jmx.auth.info` nodes are not accessible through the MBean to prevent disclosure of system credentials. Accessing the data stored at these nodes will result in a null response.
- A collective member is restricted to modifying only its own information in the repository. Authenticated administrative users have unrestricted access to information in the repository except as previously noted. Authenticated administrative users are all users granted the Administrative role.

Because the collective repository ultimately resides on the disk, the file system permission settings must be secure for the environment. It is recommended that the collective controller's configuration be readable and writable only by the user, readable only by the group, and not accessible at all by the world - in other words, `chmod 0640`. Follow any security guidelines that your organization might have established.

## Configuring a Liberty collective

You can organize Liberty servers into collectives to support clustering, administration, and other operations that act on multiple Liberty servers. By using collectives, you can efficiently and accurately deliver application services to your organization.

Distributed operating systems

IBM i

### Before you begin

The `collectiveController-1.0` feature and its capabilities are available only in WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express,



or WebSphere Application Server Liberty Core. If you have a WebSphere Application Server Liberty Network Deployment installation, you can use its `collectiveController-1.0` feature to work with collective members from WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core installations.

## About this task


A Liberty collective is a set of Liberty servers that are configured as part of the same administrative and operational domain.

Configuration and state data about a Liberty collective is housed in an active operational repository.

Membership in a Liberty collective is optional. Liberty servers join a collective by registering with a collective controller to become members. Members share information about themselves through the operational repository of the controller.

The following rules apply:

- A Liberty server can be a member of only one collective.
- Different Liberty servers on the same host can be in different collectives.
- Liberty servers on the same host that are members of a collective can coexist with Liberty servers that are not members of a collective.

 **Watch:** Introduction to creating a collective demonstrates the procedure. This video, and other information about collectives, is available on the WASdev website. [Transcript]

## Procedure

1. Create and configure your controller.
  - a. Create a server to act as the collective controller.

```
wlp/bin/server create myController
```

- b. Create the collective controller configuration.

The collective controller configuration consists primarily of the administrative domain security configuration that is used for secure communication between controllers and members.

```
wlp/bin/collective create myController --keystorePassword=controllerKSPassword
```

**8.5.5.2** By default, this collective command writes all output to a console screen. In the next step, you copy the configuration output into the `server.xml`. To write the configuration to a file instead of to a console screen, specify a `--createConfigFile=outputFilePath` parameter, for example:

```
wlp/bin/collective create myController --keystorePassword=controllerKSPassword --createConfigFile=c:\wlp\usr\servers\myController
```

After you run the **create** command, the `include` statement to use is displayed. To include the outputted file in the collective configuration, add the `include` statement to the `server.xml` file, for example:

```
<include location="c:\wlp\usr\servers\myController\collective-create-include.xml" />
```

- c. Update the `server.xml` file of the collective controller.

- Copy and paste output.

If the command wrote output to a console screen, proceed with the following steps:

- 1) Copy output from the collective command and paste it into the `server.xml` file.
- 2) Specify administrative user ID and password values for the collective. For example, change:

```
<quickStartSecurity userName="" userPassword="" />
```

to:

```
<quickStartSecurity userName="adminUser" userPassword="adminPassword" />
```

The default path for the collective controller server.xml file is `${wlp.install.dir}/usr/servers/myController/server.xml` or, if the `$WLP_USER_DIR` variable is set in a `server.env` file or command window, `$WLP_USER_DIR/servers/myController/server.xml`. After editing, the file will resemble the following example:

```
<server description="controller server">

 <!-- Enable features -->

 <featureManager>
 <feature>jsp-2.2</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080"
 httpsPort="9443" />

 <featureManager>
 <feature>collectiveController-1.0</feature>
 </featureManager>

 <!-- Define the host name for use by the collective.
 If the host name needs to be changed, the server should be
 removed from the collective and re-joined or re-replicated. -->

 <variable name="defaultHostName" value="controllerHostname" />

 <!-- TODO: Set the security configuration for Administrative access -->
 <quickStartSecurity userName="adminUser" userPassword="adminPassword" />

 <!-- clientAuthenticationSupported set to enable bidirectional trust -->

 <ssl id="defaultSSLConfig"
 keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore"
 clientAuthenticationSupported="true" />

 <!-- inbound (HTTPS) keystore -->
 <keyStore id="defaultKeyStore" password="yourPassword"
 location="${server.config.dir}/resources/security/key.jks" />

 <!-- inbound (HTTPS) truststore -->
 <keyStore id="defaultTrustStore" password="yourPassword"
 location="${server.config.dir}/resources/security/trust.jks" />

 <!-- server identity keystore -->
 <keyStore id="serverIdentity" password="yourPassword"
 location="${server.config.dir}/resources/collective/serverIdentity.jks" />

 <!-- collective trust keystore -->
 <keyStore id="collectiveTrust" password="yourPassword"
 location="${server.config.dir}/resources/collective/collectiveTrust.jks" />

 <!-- collective root signers keystore -->
 <keyStore id="collectiveRootKeys" password="yourPassword"
 location="${server.config.dir}/resources/collective/rootKeys.jks" />
</server>
```

- **8.5.5.2** Add an include statement.

If you wrote the output to a file by using the `--createConfigFile=outputFilePath` parameter, add an include statement to `$WLP_USER_DIR/servers/myController/server.xml` to include the outputted file in the collective configuration. For example:

```
<server description="controller server">

 <!-- Enable features -->

 <featureManager>
 <feature>jsp-2.2</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080"
```

```

httpsPort="9443" />
<include location="c:\wlp\usr\servers\myController\collective-create-include.xml" />
</server>

```

Ensure the outputted file sets administrative user ID and password values for the collective, for example:

```
<quickStartSecurity userName="adminUser" userPassword="adminPassword" />
```

- d. Start the collective controller server.

```
wlp/bin/server start myController
```



Figure 17. Collective of one

- e. Verify that the collective controller server started correctly and is ready to receive members.

- 1) Open an editor on the collective controller messages log, `$WLP_USER_DIR/servers/myController/logs/messages.log`.

- 2) Look for the following message:

```
CWWKX9003I: CollectiveRegistration MBean is available.
```

If you want to enable a collective controller and its members to use the security TLSv1.2 protocol for the Secure Sockets Layer (SSL) context, see “Setting up Liberty to run in SP800-131a” on page 1165. The `server.xml` files of the controller and members need `ssl id` elements and each host computer needs a `server.env` file with the `JVM_ARGS=-Dhttps.protocol=TLSv1.2` statement in its `${wlp.install.dir}/etc` directory.

2. Create and configure a member to join the collective.

The controller and members can be on separate hosts. In this example, the controller and member are on the same host.

- a. Create a member server.

```
wlp/bin/server create myMember
```

- b. Join the member.

Run the collective **join** command to join the server to the collective as a member.

The **join** command requires a network connection to the collective controller and an administrative user ID and password for performing MBean operations on the controller. Look at the `server.xml` file of the collective controller to find the values for the `--host`, `--port`, `--user`, and `--password` parameters. For `--keystorePassword`, set a value to use for the member keystore password, such as `memberKSPassword`. You can specify different `--keystorePassword` values for each server that is joined to the collective.

```
wlp/bin/collective join myMember --host=controllerHostname --port=9443 --user=adminUser --password=adminPassword --keystorePassword=memberKSPassword
```

The optional parameter `--hostname` specifies the host name to use for the system. Set `--hostname` only if the system has multiple host names or does not have its host name configured. If set, the value must match the `defaultHostName` variable that is defined in the `server.xml` file.

**8.5.5.2** To write the output of this collective command to a file, instead of to a console screen, specify an optional `--createConfigFile=outputFilePath` parameter. Then, include the outputted file in the collective configuration by adding an `include` statement to the member `server.xml` file:

#### 8.5.5.2

```
<include location=outputFilePath />
```

**8.5.5.4** By default, the **join** operation leaves remote procedure call (RPC) credentials undefined. You must specify values for `rpcUser`, `rpcUserPassword`, and the operating system login user and

password for the host on which the member server resides. If the member host is registered with the collective controller and the member host is not enabled for SSH, specify an optional `--useHostCredentials` parameter to enable the member to inherit RPC credentials from its host registration on the controller. Typically, Linux hosts are enabled for SSH and Windows hosts are not enabled for SSH; thus, the `--useHostCredentials` parameter is useful for Windows member hosts. Specifying `--useHostCredentials` adds `<hostAuthInfo useHostCredentials="true" />` to the member server.xml file. You then can run collective member server commands such as **start** or **stop** without specifying RPC credentials because the member inherits credentials from its host. See “Overriding Liberty server host information” on page 919 for information about `hostAuthInfo`, the `--useHostCredentials` parameter, and connecting the collective controller to the server.

For information about these required parameters and about optional parameters, run `collective help join` at a command line.

- c. If prompted to accept the certificate chain, enter `y` (yes).
- d. Update the member server.xml file.
  - Copy and paste output.

If the command wrote output to a console screen, proceed with the following steps:

- 1) Copy output from the collective command and paste it into the member server.xml file.
- 2) Modify the ports so that the server can open its HTTP ports. Ensure the member server.xml sets unique HTTP port numbers on its host. For example, if the member is on the same host as the collective controller, change the HTTP port numbers:

```
<httpEndpoint id="defaultHttpEndpoint" httpPort="9081" httpsPort="9444" />
```

Optionally, to access the member server from a remote client, also set `host="*"` in the `httpEndpoint` element.

In `$WLP_USER_DIR/servers/myMember/server.xml`, for example:

```
<server description="member server">
 <!-- Enable features -->
 <featureManager>
 <feature>jsp-2.2</feature>
 </featureManager>
 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9081"
 httpsPort="9444" />
 <featureManager>
 <feature>collectiveMember-1.0</feature>
 </featureManager>
 <!-- Define the host name for use by the collective.
 If the host name needs to be changed, the server should be
 removed from the collective and re-joined or re-replicated. -->
 <variable name="defaultHostName" value="memberHostname" />
```

```
8.5.5.4 <!-- Remote host authentication configuration --> <hostAuthInfo
rpcUser="admin_user_id" rpcUserPassword="admin_user_password" /> <!-- Connection to the
collective controller --> <collectiveMember controllerHost="controllerHostname"
controllerPort="9443" /> <!-- clientAuthenticationSupported set to enable bidirectional trust -->
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" trustStoreRef="defaultTrustStore"
clientAuthenticationSupported="true" /> <!-- inbound (HTTPS) keystore --> <keyStore
id="defaultKeyStore" password="yourPassword" location="{server.config.dir}/resources/
security/key.jks" /> <!-- inbound (HTTPS) truststore --> <keyStore id="defaultTrustStore"
password="yourPassword" location="{server.config.dir}/resources/security/trust.jks" /> <!--
server identity keystore --> <keyStore id="serverIdentity" password="yourPassword"
location="{server.config.dir}/resources/collective/serverIdentity.jks" /> <!-- collective truststore
```

```
--> <keyStore id="collectiveTrust" password="yourPassword" location="{server.config.dir}/resources/collective/collectiveTrust.jks" /> </server>
```

- **8.5.5.2** Add an include statement.

If you wrote the output to a file by using the `--createConfigFile=outputFilePath` parameter, add an include statement to `$WLP_USER_DIR/servers/myMember/server.xml` to include the outputted file, for example:

```
<server description="member server">
 <!-- Enable features -->
 <featureManager>
 <feature>jsp-2.2</feature>
 </featureManager>
 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9081"
 httpsPort="9444" />
 <include location="c:\wlp\usr\servers\myMember\collective-join-include.xml" />
</server>
```

- e. **8.5.5.4** If you did not specify `--useHostCredentials` in the `join` command and the member host is not enabled for SSH, set RPC credentials for `hostAuthInfo` in the `member server.xml` file or the outputted file. You can set RPC credentials for the member server in either of two ways:

- Set `hostAuthInfo` RPC user and password values. Set `rpcUser` to an operating system login user ID for the host on which the member server resides, and set `rpcUserPassword` to the operating system login password for the user ID. For example, if you log into the member computer with user `test1` and password `test1pwd`, then change the `hostAuthInfo` element to the following:

```
<hostAuthInfo rpcUser="test1" rpcUserPassword="test1pwd" />
```

- If the member host is registered with the collective controller, set `hostAuthInfo useHostCredentials` to `true` for the member server to inherit RPC credentials from its host.

```
<hostAuthInfo useHostCredentials="true" />
```

See “Overriding Liberty server host information” on page 919 for information about `hostAuthInfo` settings and for an example that shows how to register a member host and run the `join` command with `--useHostCredentials`.

- f. Start the member server.

```
wlp/bin/server start myMember
```

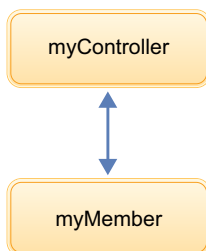


Figure 18. Simple collective

- g. Verify that the member server started correctly and is publishing information to the controller.

- 1) Open an editor on the member messages log, `$WLP_USER_DIR/servers/myMember/logs/messages.log`.
- 2) Look for the following messages in any order:

```
CWWKX8112I: The server's host information was successfully published to the collective repository.
CWWKX8114I: The server's paths were successfully published to the collective repository.
CWWKX8116I: The server STARTED state was successfully published to the collective repository.
```

## Configuring a Liberty collective using the developer tools

### 8.5.5.1

Using the Liberty **Utilities** menu in the developer tools, you can create a collective controller or join a collective.

### Before you begin

Using the developer tools simplifies the process of configuring a Liberty collective by providing convenient utilities for creating collective controllers or joining collectives. For more information about Liberty collectives, see “Configuring a Liberty collective” on page 912.

### Procedure

#### 1. Creating a collective controller:

**Note:** Creating a collective controller can be done only with the following editions:

- WebSphere Application Server Liberty Network Deployment
  - WebSphere Application Server Liberty for z/OS
- a. In the Servers view, right-click your Liberty server, and select **Utilities > Create Collective Controller**.
  - b. In the **Keystore password** field, type a password for your collective.
  - c. Click **Finish**.

Results:

An information dialog appears after the collective configuration is complete. This dialog informs you that the `server.xml` file was already updated to include the generated configuration file. If there are instructions to add the configuration lines to the `server.xml` in the console output, they can be ignored. However, Administrative Security and other settings might still need to be configured within the included configuration file before any collective members can join.

#### 2. Joining a collective controller:

**Note:** You must have a collective controller available before a server can be added to a collective. Joining a collective can be done with the following editions:

- WebSphere Application Server Liberty
  - WebSphere Application Server Liberty - Express
  - WebSphere Application Server Liberty Core
- a. In the Servers view, right-click your Liberty server, and select **Utilities > Join Collective**.
  - b. On the Join Collective page, specify the information of the collective controller you would like to join.
  - c. In the **Host** field, type in the host name for the collective controller.
  - d. In the **Port** field, type in the port number for the collective controller.
  - e. In the **User** field, type in the user name for the collective controller.
  - f. In the **Password** field, type in the password for the collective controller.
  - g. In the **Keystore password** field, type the keystore password for the collective controller if encryption was used.
  - h. Click **Finish**.

Results:

An information dialog appears after the collective configuration is complete. This dialog informs you that the `server.xml` file was already updated to include the generated configuration file. If there are instructions to add the configuration lines to the `server.xml` in the console output, they can be ignored.

## Overriding Liberty server host information

The `collectiveMember-1.0` feature enables a server to be managed by the collective controller. Most server host information can be automatically detected. In certain scenarios, however, you must provide additional host information so that the collective controller can establish a connection to the server.

**Note:** **Distributed operating systems** **IBM i** The `collectiveController-1.0` feature and its capabilities are available only in WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core. If you have a WebSphere Application Server Liberty Network Deployment installation, you can use its `collectiveController-1.0` feature to work with collective members from WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core installations.

To enable the host information override, add the following element to the `server.xml` file:

```
<hostAuthInfo rpcPort="ssh_port"
 rpcUser="user_ID"
 rpcUserPassword="password"
 rpcUserHome="user_home"
 rpcHost="host_name"
 sudoUser="sudo_user"
 sudoPassword="sudo_user_password"
 sshPublicKeyPath="public_key_path"
 sshPrivateKeyPath="private_key_path"
 sshPrivateKeyPassword="private_key_password"
```

```
8.5.5.4 useHostCredentials="true_or_false"/>
```

### **rpcPort**

This parameter specifies the port for the RPC mechanism, which is SSH port 22 by default. If your system uses a nonstandard port, set this value accordingly. If this value is not specified, the default value is 22.

### **rpcUser**

This parameter specifies the user ID that the collective controller will use to connect to the server. If the host does not support SSH or using SSH keys is not desired, you can use this parameter to specify an operating system login user. For example, if you log in to the host with the `myID` user, then you specify `rpcUser="myID"`. If this value is not specified, the default value is `System.getProperty("user.name")`.

### **rpcUserPassword**

This parameter specifies the password for the specified user ID. For example, if you log in to the host with the `myID` user and the `myPwd` password, then you specify `rpcUser="myID"` and `rpcUserPassword="myPwd"`. If this value is not specified, the server will either generate an SSH key pair or use the SSH key pair for the connection that is specified using the **privateKeyPath** and **publicKeyPath** parameters. If SSH is not installed on the server (such as on a Windows or OS/400® operating system), the password is required.

### **rpcUserHome**

This parameter specifies the home directory of the user. If this value is not specified, the default value is `System.getProperty("user.home")`. If **rpcUser** is specified, you should specify **rpcUserHome**.

## rpcHost

This parameter specifies the host on which the RPC mechanism is configured to listen. If this value is not specified, the default value is the value of the **defaultHostName** variable. If your system uses a host other than the **defaultHostName**, set this value accordingly.

## sudoUser

If this value is specified, it allows the collective controller to run commands as another, or "sudo", user instead of as the user ID used for the connection. This parameter applies only to servers that have an SSH server installed. This parameter has no default value.

## sudoPassword

This parameter specifies the password for the sudo user specified by the **sudoUser** parameter. This parameter applies only to servers that have an SSH server installed. This parameter has no default value.

## sshPublicKeyPath

This parameter specifies the path and file name of a user-specified public key file. If this value is not specified, the default is `${server.output.dir}/resources/security/ssh/id_rsa.pub`. If the specified file (or default file) does not exist, a new public key file will be generated.

## sshPrivateKeyPath

This parameter specifies the path and file name of a user-specified private key file. If this value is not specified, the default is `${server.output.dir}/resources/security/ssh/id_rsa`. If the specified file (or default file) does not exist, a new private key file will be generated.

## sshPrivateKeyPassword

This parameter specifies the password for the private key. This parameter has no default value.

### 8.5.5.4 useHostCredentials

This parameter specifies whether collective member server commands inherit RPC credentials from the host. The default is `false`, requiring the user to specify RPC credentials for the controller to remotely start or stop the member. When set to `true`, collective member server commands inherit RPC credentials from the host registration and ignore all other RPC credentials in the `hostAuthInfo` configuration element.

## Examples

Scenario 1: Server is on Windows operating system, no SSH is installed

```
<hostAuthInfo rpcUserPassword="myPassword"/>
```

Scenario 2: Server has SSH installed, SSH is running on port 2222

```
<hostAuthInfo rpcPort="2222"/>
```

Scenario 3: Need to run commands as another user

```
<hostAuthInfo sudoUser="anotherUser" sudoPassword="anotherPassword"/>
```

Scenario 4: Server is on a Windows operating system and ssh (e.g. Cygwin) is installed. With the following server configuration, the controller connects the member server with ssh. In this case, the requirement to disable Windows User Account Control (UAC) does not apply. The parameter `<user's home directory>` is the user default home directory, for example: `C:\cygwin\home\bob`

```
<hostAuthInfo rpcUserHome="<user's home directory>" />
```

### 8.5.5.4

Scenario 5: The collective controller and member are on separate hosts, and not on the same host. To specify that the member inherit RPC credentials from the host, set `useHostCredentials` to `true` in the `server.xml` file of the member. Complete the following steps to configure the member to inherit RPC credentials from the host by specifying `--useHostCredentials` in the `join` command that joins a server as a member to the collective.



1. Create, configure, and start a collective controller named `myController` as shown in step 1 of “Configuring a Liberty collective” on page 912.
2. Register the host for the member with the collective. The member and the collective controller are on different hosts.

In this scenario, the **registerHost** command uses the collective controller host `hostA.ibm.com` with port number 9443, user `admin`, and password `adminpwd`. The command registers the member host `hostB.ibm.com` with the collective, and sets `rpcUser` to an operating system login user ID for the member host `osUser1`, and `rpcUserPassword` to the operating system login password for the user ID for the member host `osUser1Pwd`. Run the **registerHost** command on the collective controller host.

```
wlp/bin/collective registerHost hostB.ibm.com --host=hostA.ibm.com --port=9443 --user=admin --password=adminpwd --rpcUser=osUser1
```

Enter `y` (yes) when prompted to accept the certificate chain. After registration, the Host `hostB.ibm.com` successfully registers. `message` displays. The collective controller host now has the operating system user ID and password of the member host.

3. On the member host, create a server named `myMember` to use as a collective member.

```
wlp/bin/server create myMember
```

4. Add the `myMember` server to the collective controller, specifying to use host credentials. In the **join** command, which is run on the member host, specify `--useHostCredentials` so that the member inherits RPC credentials from the host registration.

```
wlp/bin/collective join myMember --host=hostA.ibm.com --port=9443 --user=admin --password=adminpwd --keystorePassword=memberKSPas
```

5. Update the member `server.xml` file as shown in step 2 of “Configuring a Liberty collective” on page 912.

Because you specified `--useHostCredentials` in the **join** command, the configuration generated for the member `server.xml` file sets `useHostCredentials` to `true`:

```
<!-- Remote host authentication configuration -->
<hostAuthInfo useHostCredentials="true" />
```

With the `--useHostCredentials` option, you do not need to specify the operating system user ID and password in the member `server.xml` file because the member inherits credentials from the host. Later, if the operating system user ID or password of the member server changes, run the **updateHost** command to change the user ID or password. For more information about the **registerHost** and **updateHost** commands, see “Registering host computers with a Liberty collective.”

## Registering host computers with a Liberty collective

You can register a host computer with a Liberty collective controller, update host information, or unregister a host. Registration enables the collective controller to access applications, command files, and other resources on the host. Registered hosts are members of the collective.

### Before you begin

1. Construct a Liberty collective. See “Configuring a Liberty collective” on page 912.

#### Distributed operating systems

IBM i

The `collectiveController-1.0` feature and its capabilities are available only in multiple-server products such as WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in single-server products such as WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core. If you have a multiple-server product installation, you can use its `collectiveController-1.0` feature to work with collective members from single-server products.

2. To enable connections to servers on local and remote hosts, complete the steps for your host operating system in “Setting up RXA for Liberty collective operations” on page 925. The topic provides information about the Tivoli Remote Execution and Access (RXA) toolkit and enabling Secure Shell (SSH) protocol. You can use RXA to remotely start and stop servers, including starting and stopping servers on your local computer, and to transfer files to and from registered hosts.

**Note:** Windows In Windows, system environment variables are visible only inside the shell that RXA connects to. Setting PATH in the command window is not sufficient. You must set PATH in the system variable section of the environment variables, or use `-hostJavaHome <PATH TO IBM JAVA>` with the `updateHost` option.

3. If you want to enable a collective controller and its members to use the security TLSv1.2 protocol for the Secure Sockets Layer (SSL) context, see “Setting up Liberty to run in SP800-131a” on page 1165. The `server.xml` files of the controller and members need `ssl id` elements and each host computer needs a `server.env` file with the `JVM_ARGS=-Dhttps.protocols=TLSv1.2` statement in its `${wlp.install.dir}/etc` directory.

## About this task

A host computer is not required to have any WebSphere Application Server products installed. There are no software requirements for a host beyond its operating system. The host can be the same computer on which the product is installed or a different computer.

To register a host with a collective controller, update host information, and unregister a host, use the **registerHost**, **updateHost**, and **unregisterHost** commands. Specify the host computer name in one of the following formats:

- Fully qualified domain name servers (DNS) host name string, such as `xmachine.ibm.com`
- Default short DNS host name string, such as `xmachine`
- Numeric IP address, such as `127.1.255.3`

**Note:** When a Liberty server is joined to a collective, the associated host is automatically registered with the collective controller if it is not already registered.

A host can be registered with the collective under different names. Ensure that the host name specified for **registerHost**, **updateHost**, and **unregisterHost** is consistent with the host name used for the registered collective members. The `defaultHostName` variable in the registered server member's `server.xml` file controls the host name to which the server considers itself to belong.

## Procedure

- Register a host with a collective controller.

To register the current host where both the collective controller host and the remote target host are the same computer, run the **registerHost** command on the collective utility script with no explicit host target. Specify the collective controller's host name, port, and administrative user name and password. For example:

```
wlp/bin/collective registerHost --host=controllerHost --port=controllerHTTPSPort
--user=controllerAdmin --password=controllerAdminPassword
```

This example command generates a unique SSH key pair for authenticating to the SSH server of a specified host computer. If you are registering a remote host for which an SSH key pair is already generated, you can specify the path of the SSH private key file. The following **registerHost** command assumes that the SSH private key is stored on the local controller computer at `/home/user1/.ssh/id_rsa`. The other file in the SSH key pair is the `/home/user1/.ssh/authorized_keys` public key file on the remote target host.

```
wlp/bin/collective registerHost remotehost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
--sshPrivateKey=/home/user1/.ssh/id_rsa
```

If the remote target host does not support SSH or you do not want to use SSH keys, you can specify an operating system login user for `rpcUser` and login password for `rpcUserPassword`. If you include `rpcUser` with `rpcUserPassword`, do not include `sshPrivateKey`. The command to specify operating system login user and password resembles:

```
wlp/bin/collective registerHost remotehost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
--rpcUser=osUserForRemoteHost --rpcUserPassword=osUserPasswordForRemoteHost
```

### 8.5.5.2

To transfer files to and from a host, you must specify host read and write paths. Unless the **registerHost** command specifies the paths, you cannot deploy a Liberty archive to the host. The **hostReadPath** specifies the directories that the collective controller can read. The **hostWritePath** specifies the directories to which the collective controller can write. Paths that are specified by **hostWritePath** are also readable. For example, to upload an archive to `/opt/wlp`, you must specify `--hostWritePath=/opt`. Specify a parameter multiple times for multiple paths.

```
wlp/bin/collective registerHost myHost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
--rpcUser=osUser --rpcUserPassword=osUserPassword
--hostReadPath=/opt --hostWritePath=/dir1 --hostWritePath=/dir2
```

To use the Deploy tool of the WebSphere Liberty Administrative Center ("Admin Center"), you must set **hostWritePath** to the path to which you want to deploy a server package. To transfer files to multiple directories, include multiple instances of the **hostWritePath** parameter in the command. For example:

```
wlp/bin/collective registerHost myHost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin
--password=controllerAdminPassword --rpcUser=osUser --rpcUserPassword=osUserPassword
--hostWritePath=c:\was\liberty\brokerageAppTest --hostWritePath=c:\wlp_backup
```

Optionally, specify the path to the Java home directory of the host with the **-hostJavaHome** parameter. For example: `-hostJavaHome=c:\java\jre`

- Update registered host authentication information.

Run the **updateHost** command on the collective utility script to change the authentication information of a registered host. For example, if the user password changes, the following command updates the host password that is used by the collective:

```
wlp/bin/collective updateHost myHost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
--rpcUser=osUser --rpcUserPassword=newOsUserPassword
```

- **8.5.5.2** Update registered host read or write paths.

Run the **updateHost** command on the collective utility script to change the host read and write paths. Paths in this command override the previously set paths for **hostReadPath** and **hostWritePath**, and do not add to the existing paths.

```
wlp/bin/collective updateHost myHost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
--rpcUser=osUser --rpcUserPassword=osUserPassword
--hostReadPath=/optNew --hostWritePath=/opt --hostWritePath=/home/osUser
```

- Unregister a host from a collective controller.

Run the **unregisterHost** command on the collective utility script; for example:

```
wlp/bin/collective unregisterHost myHost.ibm.com --host=controllerHost
--port=controllerHTTPSPort --user=controllerAdmin --password=controllerAdminPassword
```

Unregistering a host removes all the registered servers on that host and any other host-based information from the collective controller.

## What to do next

For information about all parameters of the **registerHost**, **updateHost**, and **unregisterHost** commands, see the API documentation for the `CollectiveRegistration` MBean.

## Setting the `JAVA_HOME` variable for Liberty collective members

All Liberty collective members must have a Java Runtime Environment (JRE) installed that meets the minimum requirements of the Liberty server. After a JRE is installed on the host computer, you can set the `JAVA_HOME` variable so that the Liberty operation can locate the JRE.

## About this task

In order for the collective controller to perform remote operations on Windows members such as starting or stopping a member server, the collective controller must run with an IBM JRE. Third-party JREs do not contain the required security classes. You can get a JRE that supports Liberty products and SSL from Installation Manager offerings or developerWorks:

- Using Installation Manager, select the Liberty product first and then select WebSphere SDK for Liberty. Use Installation Manager to install the Liberty product and software development kit (SDK). The WebSphere SDK for Liberty includes the needed support for Liberty products and SSL and offers a Java client, JConsole.
- Go to <http://www.ibm.com/developerworks/java/jdk/index.html> on the developerWorks website and download an IBM Java development kit (JDK) for your operating system. The developerWorks website does not have a JRE for all operating systems. For example, you have to get the JDK from Eclipse for Windows operating systems.

You can set the `JAVA_HOME` variable in operating system settings or at a command line. Setting set the `JAVA_HOME` variable enables remote operations to locate the JRE.

## Procedure

- **Windows** To set `JAVA_HOME` on a Windows system, perform the following actions.

1. On the Control Panel, do the following:
  - Click **System**.
  - Click **Advanced system settings**.

The System Properties window opens.

2. Click the **Environment Variables** button.
3. Click the *New* button in the system-variables section.
4. Add the `JAVA_HOME` variable name and specify a path to the `jre` directory; for example:  
`C:\wlp_855\IBM\WebSphere\Liberty\java\java_1.7.1_64\jre`

Some collective controller commands require that the path to the Java installation `jre\bin` directory be available in the System path, so also add a path to the `jre\bin` directory.

5. Save the changes. You might need to reboot the computer for the changes to take effect.
6. To verify the changes, at a command line enter `set JAVA_HOME`. The command displays the `JAVA_HOME` settings; for example:

```
JAVA_HOME=C:\wlp_855\IBM\WebSphere\Liberty\java\java_1.7.1_64\jre
```

**Note:** Rather than change operating system settings, you can set `JAVA_HOME` at a command line by entering `set JAVA_HOME=path_to_jre`. A limitation is that the `JAVA_HOME` setting only applies to commands entered in the same command window.

- **AIX** **HP-UX** **Linux** **Solaris** If you are running bash shell, you can add the `JAVA_HOME` environment variable to the `.bashrc` file in the user's home directory.
- One option for setting `JAVA_HOME` is to create a `server.env` file in the `${server.config.dir}` where `server.xml` is residing and add `JAVA_HOME` there.

For example:

```
JAVA_HOME=/java/jre
```

- **Linux** Some collective controller commands require that the path to the Java installation `jre/bin` directory be available in the `.bashrc` file, so set a path to `jre/bin` in the `.bashrc` file.

## Setting up RXA for Liberty collective operations

Liberty collective controllers use the Tivoli Remote Execution and Access (RXA) toolkit to perform selected operations on collective members. Use RXA to remotely start and stop servers, including starting and stopping servers on your local computer.

### Procedure

- **AIX** **HP-UX** **Linux** **Solaris** **Set up Linux, UNIX or z/OS machines**

Install and enable SSH on your machine. For Linux and UNIX machines, ensure that the configuration is set according to the following instructions. For z/OS machines, consult the following instructions for guidance.

To enable SSH, configure OpenSSH 3.6.1, OpenSSH 4.7 (on AIX), or Oracle SSH 1.1 so that it supports RXA connections. OpenSSH 3.7.1 or later contains security enhancements not available in earlier releases and is recommended.

**Avoid trouble:** OpenSSH Version 4.7.0.5302 for IBM AIX Version 5.3 is not compatible with RXA Version 2.3. If machines are running AIX Version 5.3 with OpenSSH Version 4.7.0.5302 installed, file transfers might not complete. To avoid this problem, revert from OpenSSH Version 4.7.0.5302 to Version 4.7.0.5301.

### Using Secure Shell (SSH) protocol

RXA does not supply SSH code for UNIX operating systems. You must ensure that SSH is installed and enabled on all machines that include collective members.

In all UNIX environments except Solaris, the Bourne shell (sh) is used. On Solaris machines, the Korn shell (ksh) is used instead due to problems encountered with the Bourne shell (sh).

To use password-based authentication for SSH communications, edit the `/etc/ssh/sshd_config` file on each machine that includes one or more collective members. Set the `PasswordAuthentication` property to `yes`. For example:

```
PasswordAuthentication yes
```

The default value for the `PasswordAuthentication` property is `no`.

After you change this setting, stop and restart the SSH daemon by using the following commands:

```
/etc/init.d/sshd stop
/etc/init.d/sshd start
```

**Linux** Some collective controller commands require that the path to the Java installation `jre/bin` directory to be available in the `.bashrc` file, so set a path to `jre/bin` in the `.bashrc` file.

- **IBM i** **Set up IBM i machines**

Using SSH public/private key authentication to IBM i machines is not supported.

- **Windows** **Set up Windows machines**

1. Ensure that your collective controller is running with an IBM JDK.

RXA requires some security classes that are in the IBM JDK, and that are not available in the Oracle or OpenJDK JVMs.

2. Ensure the system environment variables `JAVA_HOME` and `PATH` are set to the Java path (`jre` directory) on the computer. Some collective controller commands require that the path to the Java installation `jre\bin` directory is available in the System path, so also add a path to the `jre\bin` directory.

In Windows, system environment variables are visible only inside the shell that RXA connects to. Setting `PATH` in the command window is not sufficient. You must set `PATH` in the system variable section of the environment variables, or use `-hostJavaHome <PATH TO IBM JAVA>` with the `updateHost` option.

See "Setting the JAVA\_HOME variable for Liberty collective members" on page 923.

3. Ensure that the server.xml file of each server to be managed specifies the account user name and password.

Specify the user name and password in a hostAuthInfo statement in the server.xml file:

```
<hostAuthInfo rpcUser="Windows_user_ID" rpcUserPassword="Windows_user_password" />
```

4. Enable connections to member servers on Windows computers.

To enable connections to Windows members, you can use a third-party SSH service such as Cygwin on your Windows member computer or change Windows operating system settings on a member computer that does not have an SSH service installed.

- Use a third-party SSH service such as Cygwin on the Windows member computer.

If the member computer uses an SSH service, the controller connects the member server with SSH. Specify a hostAuthInfo rpcUserHome parameter and the RPC user name and password in the member server.xml file because the third-party SSH service might have a different home directory than the one Windows uses:

```
<hostAuthInfo rpcUser="Windows_user_ID" rpcUserPassword="Windows_user_password" rpcUserHome="user_home_directory"/>
```

For *user\_home\_directory*, specify the user home for the SSH service; for example:

rpcUserHome="C:\cygwin\home\user1". The SSH public and private key pair is generated in the .ssh directory under this user home directory.

- If the Windows member computer does not use a third-party SSH service such as Cygwin, change the Windows operating system settings of the member computer to enable connections.
  - Ensure that your user account belongs to the Administrators group.

Many RXA operations require access to resources that standard user accounts cannot access. Thus, the configuration of a collective member must include the name and password of a Windows user who belongs to the Administrators group.

- Ensure File and Printer Sharing for Microsoft Networks is enabled for your network stack.
  - a. Click **Start > Control Panel > Network and Sharing Center > Change advanced sharing settings**.
  - b. Select **Turn on file and printer sharing**.
  - c. Save the changes.

Ensure that file sharing operations (on port 445) are not blocked on machines that include collective controllers or collective members. For more information, see the documentation for your operating system or your firewall software.

- Start the Remote Registry service.

The Remote Registry service must be running on computers that include collective members for the collective controllers to remotely run commands and scripts.

- a. Click **Start > Administrative Tools > Services**.
- b. Within the list of services, locate the Remote Registry entry and verify that the status is **Started**. If you intend to use RXA regularly, consider setting the Remote Registry **Startup type** property to **Automatic**.

- Disable User Account Control.
  - a. Click **Start > Control Panel > User Accounts > Change User Account Control settings**.
  - b. Set the User Account Control level to **Never notify**.
  - c. Click **OK**.
  - d. Restart the computer for the changes to take effect.

For more information, see Liberty Collectives Remote Operation Configuration.

## What to do next

If you modified the server.xml of a managed server, manually start the server so that it publishes the new data to the controller.

After you enable RXA, test the host configuration and verify RXA connectivity:

---

## Setting up a Liberty server to use Bluemix services

8.5.5.9

You can configure a Liberty server to use Bluemix services. Not all Bluemix services are available for configuration.

### bluemixUtility command

8.5.5.9

Use the IBM Bluemix command-line utility to configure your on-premises Liberty server to use certain Bluemix cloud services.

### Sources

Learn about Bluemix services, such as Watson and Cloudant<sup>®</sup> services, that you can use with the command-line utility.

- To register for or log in to a Bluemix account, see [Sign up for IBM Bluemix](#).
- To learn more about Watson, see [Watson services](#).
- To learn more about Cloudant services, see [Getting started with Cloudant NoSQL DB](#).

### Syntax

The command syntax is as follows:

```
bluemixUtility action [options]
```

Use the following *action* commands:

**login** Log in to Bluemix. If you run the **login** command without any options, the tool prompts you for more information, such as username and password. After a successful login, Bluemix credentials are saved to a file so that you can run other commands without specifying username and password again.

If you log in to Bluemix with the Cloud Foundry (cf) client, the bluemixUtility uses the credentials saved by the cf client. For more information, see [Cloud Foundry \(cf\) commands](#).

**marketplace**

List all the Bluemix services that can be configured by using the command-line utility.

**createService**

Create a service instance from the Bluemix catalog.

**listServices**

List all the available Bluemix service instances.

**showService**

Show information about a service instance.

**import**

Import a configuration for a service. The imported service configuration and its dependencies are placed in the following directory:

```
${wlp.user.dir}/shared/config/services/serviceName
```

**listImports**

List all imported service configurations that can be bound to a Liberty server.

**bind**

Bind a Bluemix service configuration to a Liberty server. The configuration for a service may provide default values for certain options, such as the `jndiName` of a `dataSource` element. In some

cases, the default values do not match what the application expects. Use the `--v` option to override a default value with the value that your application expects.

#### **unbind**

Unbind a service configuration from a Liberty server.

#### **deleteService**

Delete a service instance.

**switch** Switch to a different Bluemix organization or space.

**info** View Bluemix connection information.

**help** Use the help action on each command to view descriptions, usage, and options.

**logout** Log out of Bluemix. The **logout** command deletes the file that was created when you logged in with Bluemix credentials.

## **Usage**

View usage examples that you can run for each action.

Use the following command to run the **login** action:

```
bluemixUtility login [options]
```

Use the following command to run the **marketplace** action:

```
bluemixUtility marketplace [serviceType...]
```

Use the following command to run the **createService** action:

```
bluemixUtility createService [options] serviceType servicePlan serviceName
```

Use the following command to run the **listServices** action:

```
bluemixUtility listServices
```

Use the following command to run the **showService** action:

```
bluemixUtility showService [options] serviceName
```

Use the following command to run the **import** action:

```
bluemixUtility import [options] serviceName
```

Use the following command to run the **listImports** action:

```
bluemixUtility listImports [serverName]
```

Use the following command to run the **bind** action:

```
bluemixUtility bind [options] serverName serviceName
```

Use the following command to run the **unbind** action:

```
bluemixUtility unbind serverName serviceName
```

Use the following command to run the **deleteService** action:

```
bluemixUtility deleteService [options] serviceName
```

Use the following command to run the **switch** action:

```
bluemixUtility switch [options]
```

Use the following command to run the **logout** action:

```
bluemixUtility logout
```



## Options

View the available options for each action.

The following options are available for the **bluemixUtility login** command:

**--api=url**

Bluemix API endpoint, for example, `https://api.ng.bluemix.net`. The API endpoint can also be set as the Bluemix region name. For example, it can be set to `us-south` for the US South region, `eu-gb` for the London, UK region, and `au-syd` for the Sydney, Australia region.

**--user=username**

User name of Bluemix account.

**--password=password**

Password of Bluemix account.

**--org=organizationName**

Organization name.

**--space=spaceName**

Space name.

The following options are available for the **bluemixUtility marketplace** command:

**[serviceType...]**

Show detailed information about a particular Bluemix service. Specify multiple service names by separating them with a space.

The following options are available for the **bluemixUtility createService** command:

**--credentialName=name**

The name of the service credential. By default, `credential-1` is used.

**serviceType**

The type of service to create.

**servicePlan**

The name of service plan.

**serviceName**

The name of service to create.

No options are available for the **bluemixUtility listServices** command.

The following options are available for the **bluemixUtility showService** command:

**--showCredentials**

Display service credentials.

**serviceName**

The name of a Bluemix service.

The following options are available for the **bluemixUtility import** command:

**--acceptLicense**

Automatically indicate acceptance of license terms and conditions.

**--credentialName=name**

The name of the service credential. By default, the first credential that is found is used.

**--encodeAlgorithm=[xor|aes]**

Specifies how to encode sensitive information in the imported service configuration. Supported encodings values are **xor** and **aes**. The default encoding algorithm is **xor**.

**--encodeKey=key**

Specifies the key to be used when you are encoding with AES encryption. If this option is not provided, a default key is used.

**--p[parameter]=value**

Specifies parameters that help in generating and importing a configuration for a service.

**serviceName**

The name of a Bluemix service.

The following options are available for the **bluemixUtility listImports** command:

**[serverName]**

List the services that are already bound to this particular server.

The following options are available for the **bluemixUtility bind** command:

**--v[variable]=value**

Override variables in the imported service configuration.

**--acceptLicense**

Automatically indicate acceptance of license terms and conditions.

**serverName**

Name of the server to bind to the service configuration.

**serviceName**

Name of the imported service configuration.

The following options are available for the **bluemixUtility unbind** command:

**serverName**

Name of the server to unbind the service configuration from.

**serviceName**

Name of the service configuration to unbind.

The following options are available for the **bluemixUtility deleteService** command:

**--force** Force deletion without confirmation.

**serviceName**

Name of the service to delete.

The following options are available for the **bluemixUtility switch** command:

**--org=organizationName**

Organization name.

**--space=spaceName**

Space name.

No options are available for the **bluemixUtility info** command.

No options are available for the **bluemixUtility logout** command.

## Return codes

Table 74. Return codes and explanations

Return code	Explanation
0	The command successfully completed the requested operation.
20	One or more command-line arguments or options are not valid.
21	An unknown runtime exception has occurred.
22	An IO error occurred, typically when trying to delete a file from the file system.
24	User abort. Occurs when user fails to respond to a prompt or cancels the operation.
26	An unknown exception has occurred.
27	Bluemix authentication error when attempting to log in, or attempting to perform a task without being logged in.
28	A generic error when communicating with Bluemix.
29	A generic error when communicating with the configuration service.
30	A generic service configuration error has occurred.
31	A generic feature installation error has occurred.
255	An unknown error has occurred.

## Configuring Liberty for Bluemix Cloudant services

8.5.5.9

Use the IBM Bluemix utility command-line integration tool to configure your Liberty server to use the Bluemix Cloudant service.

### Before you begin

Before you configure your Liberty server to use Bluemix services, you must create an account. See Sign up for IBM Bluemix to create your Bluemix account.

### About this task

Configure your Liberty server to use the Cloudant service. For more information about Cloudant, see Getting started with Cloudant NoSQL DB.

### Procedure

1. Log in using the **bluemixUtility login** command. After your initial login, you do not have to complete this step again.
2. Run the **bluemixUtility marketplace** command to list details about all the Bluemix services that can be used with the command-line utility. See the following example:

Service: cloudantNoSQLDB

Description: Cloudant NoSQL DB is a fully managed data layer designed for modern web and mobile applications that leverages Apache CouchDB. Cloudant is built upon and compatible with Apache CouchDB and accessible through a secure HTTPS API, which scales as needed. Cloudant is ISO27001 and SOC2 Type 1 certified, and all data is stored in triplicate across separate physical nodes in multiple data centers.

Documentation: <https://console.ng.bluemix.net/docs/#services/Cloudant/index.html#Cloudant>

Plans: Standard, Lite

- 3.

**Note:** If you already created an instance, skip to step 4. You can also create an instance from the Bluemix dashboard.

Run the `bluemixUtility createService [options] serviceType servicePlan serviceName` command to create an instance of the Bluemix services that you want to use. See the following example and description of a service listed:

```
bluemixUtility createService cloudantNoSQLDB Lite myCloudantService
```

- Optional: Run the **bluemixUtility listServices** command to view the Name, Type, and Plan of all the services instances that you created. See the following example:

```
myCloudantService cloudantNoSQLDB Lite
```

- Run the **bluemixUtility import myCloudantService --pversion=v2** command to import the configuration.

**Important:** The `--pversion=v2` option installs the `cloudant-1.0` feature and downloads the official Cloudant library for Java. For API information, see the official Cloudant Java library API documentation. If you omit the `--pversion=v2` option, the `couchdb-1.0` feature is installed and Ektorp libraries are used instead to communicate with the Cloudant instance.

- Accept the license terms and conditions of the necessary libraries that are needed to access the service.
- After the configuration is successfully imported, complete any additional steps to use the imported configuration in your application, such as adding a classloader reference to the library. For example, you must add the following classloader reference to your application to use the downloaded libraries.

```
<application id="myCloudantApp">
 <classloader commonLibraryRef="cloudantNoSQLDB-library"/>
</application>
```

- Optional: Run the **bluemixUtility listImports** command to view the service configurations that you imported.

The following IBM Bluemix service configurations have been imported:  
myCloudantService

- Run the `bluemixUtility bind [options] serverName serviceName` to bind the configuration to a Liberty server. Accept the license terms and conditions, if you are prompted. See the following example and description of a service listed:

```
bluemixUtility bind defaultServer myCloudantService
```

```
Checking if features required for the myCloudantService are installed.
All required features are installed.
The myCloudantService is now bound to defaultServer server.
```

If you imported the service with the `--pversion=v2` option to use the official Cloudant Java libraries, the default JNDI name for the Cloudant database is `cloudant/serviceName`. If you did not specify that option, the default JNDI name is `couchdb/serviceName`. If your application references the database by using a different JNDI name, use `--vjndiName` option to specify the JNDI name.

```
$ bluemixUtility bind defaultServer myCloudantService --vjndiName=couchdb/connector
```

## Results

You can now use Cloudant services with your Liberty server.

## Configuring Liberty for Bluemix Watson services

8.5.5.9

Use the IBM Bluemix utility command-line integration tool to configure your on-premise Liberty server to use Bluemix Watson services.

## Before you begin

Before you configure your Liberty server to any Bluemix service, you must create an account. See

Sign up for IBM Bluemix to create your Bluemix account.

## About this task

Configure your Liberty server to use Watson services. Your application must use the Watson Developer Cloud Java SDK to access the Watson service. For more information, see Watson Developer Cloud Java SDK services. For more information about Watson, see Watson services.

## Procedure

1. Log in using the **bluemixUtility login** command. After you initially log in, you do not have to complete this step again.
2. Run the **bluemixUtility marketplace** command to list details about all the Bluemix services that can be used with the command-line utility. See the following example of the Concept Insights description:

Service: personality\_insights

Description: The Watson Personality Insights derives insights from transactional and social media data to identify ps

Documentation: <https://www.ibm.com/watson/developercloud/personality-insights.html>

Plans: tiered, premium

3.

**Note:** If you already created an instance, skip to step 4. You can also create an instance from the Bluemix dashboard.

Run the `bluemixUtility createService [options] serviceType servicePlan serviceName` command to create an instance of the service that you want to use. You can choose a unique `serviceName` for your service instance. The following example creates an instance of the Personality Insights service:

```
bluemixUtility createService personality_insights premium myWatsonService
```

4. Optional: Run the **bluemixUtility listServices** command to view the Name, Type, and Plan of all the service instances that you created. See the following example:

```
myWatsonService personality_insights premium
```

5. Run the **bluemixUtility import myWatsonService** command to import the configuration.
  - a. Accept the license terms and conditions of the necessary libraries that are needed to access the service.
  - b. After the configuration is successfully imported, complete any additional steps to use the imported configuration in your application, such as adding a classloader reference to the library. The classloader element must be added to each application that is using the service. The classloader reference ID is different from each unique Watson service. For example, you must add the following classloader reference to your application to use the Watson Java SDK libraries.

```
<application id="myWatsonApp">
 <classloader commonLibraryRef="personality_insights-library"/>
</application>
```

6. Optional: Run the **bluemixUtility listImports** command to view the service configurations that you imported.

The following IBM Bluemix service configurations have been imported:

```
myWatsonService
```

7. Run the `bluemixUtility bind [options] serverName serviceName` to bind the configuration to a Liberty server. See the following example and description of a service listed:

```
bluemixUtility bind defaultServer myWatsonService
```

Checking if features required for the myWatsonService are installed.

All required features are installed.

The myWatsonService is now bound to defaultServer server.

## Results

You can now use Watson services with your Liberty server.

---

## Platform-as-a-service environment considerations for setting up Liberty

Platform-as-a-service (PaaS) environments, such as IBM Bluemix, Pivotal Cloud Foundry, and OpenShift Enterprise, provide management and monitoring of application instances, but they also have some restrictions. Because of the inherent characteristics of PaaS environments, some Liberty features are redundant or behave differently, and they are therefore not supported.

### Liberty server management restrictions

Features related to Liberty collectives do not apply to a PaaS environment because all Liberty server JVM instances are started, stopped, and managed by the PaaS infrastructure. The Liberty Admin Center feature is not designed to be used in a PaaS environment, where an application can be scaled to use multiple JVM instances without a collective controller. In this topology, a request to Admin Center could be directed to any of the running instances and have visibility only to the server on which the request runs.

The following administrative features are not supported in a PaaS environment:

- adminCenter-1.0
- clusterMember-1.0
- collectiveController-1.0
- collectiveMember-1.0
- dynamicRouting-1.0
- healthAnalyzer-1.0
- healthManager-1.0
- scalingController-1.0
- scalingMember-1.0

### File system restrictions

Most PaaS environments do not provide a persistent local file system to their applications. For Liberty, this impacts both the applications and the components within the server that write data locally and expect it to persist across a server JVM restart.

The Liberty transaction manager writes log files to the local file system when multiple resource managers are involved in the transaction. If the logs are not available after a JVM failure and restart, then transactions cannot be automatically completed and must be manually resolved to unlock data and to make it consistent across resource managers. To avoid this scenario, the Liberty buildpack or cartridge prevents transaction log records from being written and raises an exception to the application to prevent the second resource from enlisting. As a result, although you can still use transactions with a single XA resource, a second transactional resource cannot be enlisted in a transaction. Additionally, Web Services Atomic Transactions cannot be used because they always write log records.

If your PaaS environment provides persistent storage, then you can modify the Liberty buildpack or cartridge to enable two-phase transactions by removing the following Java property from the JVM configuration:

```
-Dcom.ibm.tx.jta.disable2PC=true
```

The following features depend on persistent local storage:

- wsAtomicTransaction-1.2
- Other features that use transactions, depending on the application behavior

## Network restrictions

In general, PaaS routers do not support Internet Inter-ORB Protocol (IIOP) traffic, so remote requests to Enterprise JavaBeans (EJB) components cannot be used. The following features depend on IIOP transport:

- appClientSupport-1.0
- appSecurityClient-1.0
- ejbRemote-3.2

In some situations, such as SSL termination at the router, Liberty relies on HTTP headers to describe aspects of the original client request. When you use SSL in a PaaS environment, the headers must be set by the PaaS router. On IBM Bluemix, these headers are already set, so you can use the `ssl-1.0` feature and any features that depend on it without changes. To get the expected behavior in other PaaS environments, you might need to configure the router to set these headers as described in NGINX and WebSphere Application Server.

The following features require the router to set HTTP headers:

- `ssl-1.0`
- Other features that depend on `ssl-1.0`, as listed in the *Features that enable this feature* section of “Secure Socket Layer” on page 568





---

## Chapter 5. Administering Liberty

A server configuration consists of a `server.xml` file, a `bootstrap.properties` file, and any optional files that are included by the two main configuration files. You can use WebSphere Application Server Developer Tools for Eclipse or a text editor to edit the configuration files. There is no administrative console for Liberty, but you can use Admin Center to administer Liberty servers and applications and other resources from a web browser on a smartphone, tablet, or computer.

### About this task

Liberty is configured by exception. The runtime environment operates from a set of built-in configuration default settings, and you only need to specify configuration that overrides those default settings. You do this by editing either the `server.xml` file or another XML file that is included in `server.xml` at run time.

Features are the units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server. They are the primary mechanism that makes the server composable. The list of features that you specify in the server configuration provides a functional server.

When you first install and start the server, a feature manager and a default server configuration are available:

- By default, a server contains the `jsp-2.2` feature, to support servlet and JSP applications. You can use the feature manager to add the features that you need.
- Server configuration is by exception. When you specify the features that you need, the default configuration of those features provides a rich environment that is designed to cover most common requirements, therefore you only need to specify changes from the default configuration.

You can organize Liberty servers into collectives to administer multiple Liberty servers at a time efficiently and accurately. For information about collectives, see “Setting up the server-management environment for Liberty by using collectives” on page 907.

### Procedure

- \*\*\*\* MISSING FILE \*\*\*\*
- “Administering Liberty by using developer tools” on page 938
- “Administering Liberty manually” on page 946
- Chapter 6, “Extending Liberty,” on page 1095

---

## XML escape characters

You need to escape specific XML characters in Liberty configuration files, such as the `server.xml` file because Liberty does not automatically escape these characters. If you use the Rational Application Developer tool, you do not need to manually escape these characters.

*Table 75. XML characters.* For more information, see Extensible Markup Language (XML) .

Original character	Escaped character
"	&quot;
'	&apos;
<	&lt;
>	&gt;
&	&amp;

---

## Administering Liberty by using developer tools

### Distributed operating systems

You can modify how the workbench interacts with Liberty by using the server editor.

#### Procedure

1. In the Servers view, right-click the server and select **Open**.
2. The server editor opens.

#### What to do next

You can modify the publishing, timeout, and other settings regarding the interaction between the workbench and the server.

## Editing the Liberty configuration by using developer tools

### Distributed operating systems

You can modify the behavior of Liberty by editing the configuration. For example, you can configure which HTTP ports to use, what features are enabled, and logging and tracing settings.

#### Before you begin

For a description of the underlying process of configuring a server, and detailed information about specific aspects of server configuration, see “Administering Liberty manually” on page 946.

#### About this task

The server configuration editor consists of two views: the Source view and the Design view. The Design view is a structured view of the file whereas the Source view is a text view. But both of them are views of the same server configuration file. The Design view has many features that help with the construction of some of the more complex elements in the server configuration such as data sources.

The following steps are demonstrated in the Design view.

#### Procedure

1. To open the Server Configuration editor, select any of the following options:
  - In the **Servers** view, right-click the server configuration and select **Open**.
  - In the **Enterprise Explorer** view, expand your server project and the server folder. Right-click the `server.xml` file and select **Open**.
2. Under the **Configuration Structure** section, the elements in the configuration are displayed.
3. Under the **Feature Manager** section, the details for the currently selected element are displayed. The details can also be modified here.
4. To add new elements, select **Server Configuration** under the **Configuration Structure** section then click **Add**.
5. To add child elements, select the parent element under the **Configuration Structure** section and then click **Add**.
6. You can remove or move elements by selecting the element and using the **Remove**, **Up**, and **Down** buttons.

## What to do next

You can use the key combinations from Table 76 in both the Source view and the Design view.

Table 76. Keyboard shortcuts

Key combination	Function
Ctrl+Space	Content assistance. It displays variables that have the correct type for the current field (int, string, and other code) or ids of the correct type for a reference field.
F3	Hyperlink to variable references and id references.

## Specifying the Liberty configuration with dropins files by using developer tools



8.5.5.6

You can specify the server configuration for Liberty by creating configuration dropins files and placing these files in the configDropins directory.

### About this task

You can use the dropins files to set a default configuration, or override configurations that are set in the server.xml file or included files. The server configuration validator includes configuration dropins files as part of the overall server configuration validation. The server configuration editor shows variables and references that are defined in dropins files in the appropriate content assistance and drop-down lists.

You can place configuration dropins files in either the `${server.config.dir}/configDropins/defaults` directory or in the `${server.config.dir}/configDropins/overrides` directory.

- If you place these files in the defaults directory, then the configuration is applied before the server configuration. In this case, the files provide default values, which you can override in the main server.xml file or the included files.
- If you place the configuration dropins files in the overrides directory, then the configuration is applied after the server configuration. In this case, the files override the main server.xml or included files.

For more information about configuration dropins files, see: “Using the configuration dropins folder to specify server configuration” on page 973.

### Procedure

1. To create a configuration dropins file, right-click your server in the Servers view.
2. Complete one of the following options:
  - a. To create a configuration dropins file in the defaults directory, select **New > Configuration Dropins File > Defaults**
  - b. To create a configuration dropins file in the overrides directory, select **New > Configuration Dropins File > Overrides**.
3. Enter the name of the file that you want to create.
4. Click **OK**.

The new file is displayed.
5. To see the merged configuration, complete one of the following options:
  - a. Right-click the server.xml file in the Enterprise Explorer or Project Explorer view, and then select **Liberty > Open Merged View**.
  - b. In the Servers view, right-click on **Server Configuration** and select **Open Merged View**.

The merged view of the server configuration shows the defaults at the beginning of the merged configuration. It also shows the overrides at the end of the merged configuration.



## Starting and stopping a server by using developer tools


You can start and stop a server by using the Liberty developer tools.

### Procedure

- To start a server:

In the Servers view, right-click the server and click **Start** to start the server or click **Debug** to start the

server in the debug mode. Alternatively, click  to start the server, or click  to start the server in the debug mode.

**Tip:**  8.5.5.4 When you start a remote Liberty server, you can specify which debug port to use in the **Remote Start Settings** section of the server editor. The default port is 7777.

- To stop the server:

In the Servers view, right-click the server and click **Stop**. Alternatively, click  to stop the server.

**Tip:** In the Servers view, you must select the server entry to perform these tasks. Do not select the server configuration entry, for example Server Configuration [server.xml], to perform these tasks.

### What to do next

You can configure the server elements by using the tools.

## Switching a Liberty Docker server between run and debug mode by using developer tools

8.5.5.9

For your Liberty Docker server, you can use WebSphere Developer Tools to switch your server between run and debug mode. When you switch between these modes, the tools create a Docker image and a Docker container, which you can save.

### Before you begin

You must create a Liberty Docker server by using WebSphere Developer Tools. For more information about using WebSphere Developers Tools to create a Liberty server in a Docker container, see “Creating a workbench Liberty server in a Docker container by using developer tools” on page 890.

### About this task

When you create a Liberty server in a Docker container, the run command and host mapped ports are fixed. As a result, when you switch a Liberty Docker server between run and debug modes, WebSphere Developer Tools commit a new Docker image based on your Docker container. This action preserves all of your changes to the container at that point. Then, the tools create a new container from the new image, but modify the run command to either run or debug the server. When you switch to debug mode, the tools map the debug port to the host.

The tools do not delete the original image and container. However, the tools remove any temporary images and containers when you switch modes. To avoid losing changes that you made to your server in debug mode, you can save the last temporary image and container when you delete a Liberty Docker server.

## Procedure

1. Switch your server from run mode to debug mode:
  - a. Open the Servers view in WebSphere Developer Tools by clicking **Window > Show view > Servers**.
  - b. Start your server in debug mode by right-clicking *your\_server* and selecting **Debug**.  
The tools go through the following steps:
    - 1) Disconnect from the *your\_server* container and stop the container.
    - 2) Commit the *your\_server* container to a new image that is named *your\_server\_debug\_websphere-liberty*
    - 3) Create a new container that is named *your\_server\_debug* from the *your\_server\_debug\_websphere-liberty* image.
    - 4) Connect to the *your\_server\_debug* container.
  - c. If the tools display the Server Execution Mode Switch window, click **OK** to continue.  
The tools display the Server Execution Mode Switch window only the first time that you switch the mode for your server.
2. Switch your server back to run mode from debug mode by right-clicking *your\_server* and selecting **Run**.  
The tools go through the following steps:
  - a. Disconnect from the *your\_server\_debug* container and stop the container.
  - b. Commit the *your\_server\_debug* container to a new image that is named *your\_server\_run\_websphere-liberty*
  - c. Create a new container that is named *your\_server\_run* from the *your\_server\_run\_websphere-liberty* image.
  - d. Connect to the *your\_server\_run* container.
3. Delete the *your\_server* server by right-clicking *your\_server* and selecting **Delete**.
4. Click **Yes** to save the *your\_server\_run* container and the *your\_server\_run\_websphere-liberty* image.

## Results

You switched your server from run mode to debug mode and back to run mode. When you switched back to run mode, you saved any changes that you made to the container and image when you were in debug mode.

## Defining a utility project as a shared library

### Distributed operating systems

You can define a utility project as a shared library and associate defined shared libraries with an application or web project.

### Before you begin

To use the shared library function in the workbench, you must create a utility project and define it as a shared library. The utility project is the only project type that can be used as a shared library.

### About this task

A shared library is an external Java archive (JAR) file that is used by one or more applications. Using shared libraries enables multiple application published on a server to use a single library, rather than use multiple copies of the same library. After you associate shared libraries with an application or project, the

application or module class loader loads classes in the shared libraries and makes those classes available to the application or module.

## Procedure

To define a utility project as a shared library:

1. Create a utility project:
  - a. In the toolbar, select **File > New > Project**.
  - b. Expand **Java EE** and select **Utility Project**. Click **Next**.
  - c. In the **Project name** field, specify a name for the utility project.
  - d. Under the **Ear membership** section, clear the **Add project to an EAR** check box.
  - e. Under the **Target runtime** section, verify the WebSphere Application Server Liberty is selected.
  - f. Click **Finish**.
2. Define the artifacts in the newly created utility project. For example, you can add Java classes to the utility project.
3. Define the utility project as a shared library:
  - a. In the **Project Explorer** view, right-click the utility project and select **Properties > Liberty > Shared Library**.
  - b. In the **Shared library ID** field, type a string as an identifier for the shared library.
  - c. In the **Archive directory** field, type or browse to a directory where you want to place the compressed copy of your utility project as a JAR file. The file name convention of the JAR file is *utilityProjectName.jar*, where *utilityProjectName* is the name of the utility project.
  - d. In the **Liberty Shared Library** page, click **Apply** to confirm your changes. Click **OK** to close the Properties window.
4. Add the utility project to the server. **Distributed operating systems** For more details see “Adding and running an application on Liberty by using developer tools” on page 1331 topic.

## Results

Here is an example entry added to the server configuration (server.xml) file:

```
<library id="libid">
 <fileset dir="C:\temp" includes="Util.jar"/>
</library>
```

In addition, the JAR file is added in the specified archive directory. In the previous example, the *Util.jar* file is added in the *C:\temp* directory.

## Setting a web project to use shared libraries

### Distributed operating systems

If you have a utility project defined as a shared library, you can associate defined shared libraries with a web project.

### Before you begin

- Define a utility project as a shared library.

### About this task

A shared library is an external Java archive (JAR) file that is used by one or more applications. Using shared libraries enables multiple application published on a server to use a single library, rather than use multiple copies of the same library. After you associate shared libraries with an application or project, the

application or module class loader loads classes in the shared libraries and make those classes available to the application or module.

## Procedure

1. To set a web project to use shared libraries:
  - a. In the **Project Explorer** view, right-click your web project that you want to associate shared libraries.
  - b. Select **Properties > Liberty > Shared Libraries**.
  - c. When you click the **Add** button, a list of shared library **IDs** will appear. In the **IDs** field, specify one or more shared library identifiers that you want the project to reference. To specify multiple identifiers, use a comma-separated list. For example: ID1, ID2, ID3

**Tip:** The shared library identifier is the value specified in the **Shared Library ID** field from the Defining a utility project as a shared library task.

2. You might want to add its associating utility projects to the class path for compilation-purpose:
  - a. In the **Project Explorer** view, right-click your project that you are associating shared libraries.
  - b. Select **Properties > Java Build Path**.
  - c. Select the **Projects** tab.
  - d. Click **Add**.
  - e. Select the utility projects that the project references.
3. Develop the artifacts in the web project. For example, you can add a servlet in a web project that references classes in the shared libraries.
4. Add the web project to the server. **Distributed operating systems** For more details see “Adding and running an application on Liberty by using developer tools” on page 1331 topic.

## Results

Here is an example entry added to the server configuration (server.xml) file:

```
<application type="war" id="web" name="web" location="web.war">
 <classloader commonLibraryRef="libid"/>
</application>
```

## Exploring the runtime environment by using developer tools

### Distributed operating systems

You can use the **Runtime Explorer** view to browse the available servers. This view shows all of the available servers for the runtime environment as opposed to the **Servers** view, which shows only those servers that are configured in the workspace. In both the Runtime Explorer and Servers view, you can expand each server to show the configuration for that server.



## Before you begin

The application server that the Runtime Explorer view supports is the WebSphere Application Server Liberty.

## About this task

You can use the Runtime Explorer view to complete the following tasks:

- View the servers that are defined in a runtime environment.
- View the shared configurations files that are defined in a runtime environment.

- Create a server in a runtime environment.
-  **8.5.5.2** Create a runtime environment and optionally install add-ons from the Liberty Repository on the Downloads page on WASdev.net.
  -  **8.5.5.6** Configure additional repositories.

## Procedure

To open the Runtime Explorer view:

1. In the toolbar of the workbench, select **Window > Show View > Other**.
2. In the **Show View** window, expand **Server** and select **Runtime Explorer**. Click **OK**.

## Displaying the server configuration in a merged view

### Distributed operating systems

You can use the **Merged Configuration** view to see a flattened view of the server configuration and any included configuration files.

### About this task

In the Server Configuration editor, under the **Configuration Structure** section, you can use the **Include** element to import files that contain additional configuration settings. The **Include** element can embed multiple layers of configuration files within the `server.xml` file, which can make the configuration difficult to read without tools. The **Merged Configuration** view provides a flattened view of the server configuration and any included configuration files. This is a read-only view and cannot be edited.

## Procedure

To display the server configuration in a merged view:

In the Servers view, right-click the server configuration and select **Open Merged View**.

## Example

Here is an example of the source code for the `server.xml` file. Look at the **include** tag which imports the `common.xml` file:

```
<server>
 <featureManager>
 <feature>jsp-2.2</feature>
 <feature>servlet-3.0</feature>
 </featureManager>
 <application id="Web2.5" location="Web2.5.war" name="Web2.5" type="war"/>
 <include location="common.xml"/>
</server>
```

Here is an example of the source code for the `common.xml` file:

```
<server>
 <application id="Setup" location="Setup.war" name="Setup" type="war"/>
</server>
```

Look under the **Configuration Structure** sections to see the difference between the Server Configuration editor and Merged Configuration view. The Merged Configuration view replaces the **Include: common.xml** element from the Server Configuration editor with **Application: Setup** element.



## Viewing the schema documentation for the server configuration

### Distributed operating systems

You can view the schema documentation for the server configuration (server.xml file) within the workbench. The documentation provides information about the configuration elements that are available, the default settings, and details for each of the elements.

### Procedure

In the **Servers** view, right-click on the server configuration and select **Open Schema Reference**.

## Generating a Liberty server dump using developer tools

Using the Liberty Utilities menu, you can generate a server dump for support.

### About this task

You can use the Utilities menu of your Liberty server to generate a server dump to capture all service-related information for support.

### Procedure

1. In the Servers view, right-click your Liberty server, and select **Utilities > Generate dump for Support...**
2. In the **Generate Dump** page, enter a location for the dump archive or use the **Browse** button. Then, click **Finish** to create the server dump file.
3. In the Console view, the path to the server dump is indicated:

```
Server TestServer dump complete in
D:\LibertyUnzip\wlp\usr\servers\TestServer\TestServer.dump-12.03.22_15.00.11.zip.
```

## Packaging a Liberty server by using developer tools

You can create a compressed file containing a server runtime environment, server configuration, and applications using the packaging wizard.

### About this task

Because a Liberty server is lightweight, you might find it useful to package up your applications and server in a compressed file. You can then store this package, distribute it to colleagues, use it to deploy the application to a different location or to another machine, or even embed it in your product distribution.

### Procedure

1. In the Servers view, stop the server.
2. Right-click your Liberty server, and select **Utilities > Package Server...**
3. In the Package Server page, in the **Archive** field, type a filename and path for your archive package, or click **Browse** to locate a filename and path. This filename can include a full path name. If the full path is omitted, a compressed file called *package\_file\_name.zip* is created in the  $\{\text{server.output.dir}\}$  directory.
4. In the **Include** field, select whether to include:
  - **Full runtime (all)**
  - **Minimal runtime (minify)**
  - **No runtime (usr)**

For details on these include options, see *Packaging a Liberty server from the command prompt* topic.

5. Click **Finish**.

## Adding a data source by using developer tools

You can add a data source to your application using the developer tools.

### Procedure

1. In the Project Explorer view, expand *liberty\_name* > **servers** > *server\_name* > **server.xml**, and select **Open**.
2. In the design view, select **Server Configuration**, and click **Add**.
3. On the Select item to add to Server Configuration: page, select **Data Source** and fill in the JNDI name.
4. On the Data Source Details panel, alongside **JDBC Driver**, click **Add**.
5. On the Configure a JDBC driver page,
  - In the **Library ref:** field, select an option from the drop-down list, or click **Add** to create a new one.
    - In the **Fileset ref:** field, select an option from the drop-down list, or click **Add** to create a new one.
      - In the **Base directory** field, type a path for your base directory or click **Browse** to locate a library location path. Use the drop-down arrow next to Browse to choose between browsing for a relative path or browsing for an absolute path.
      - In the **Includes** field, type the name of your data source, for example *derby.jar*, or click **Browse** to locate the data source archive file that you want to include.
      - In the **Excludes** field, type the archives that you want to exclude, or click **Browse** to locate the data source archive file that you want to exclude.
6. Your JDBC data source has been added to your application.

---

## Administering Liberty manually

You can administer Liberty from the command prompt, configure it with web server plug-ins, and capture its status. You can package a Liberty server configuration along with the applications that it runs, for distribution to colleagues, or installation on other systems. If available, you can use the Equinox OSGi console to aid with debugging.

### About this task

The *server.xml* file is the primary configuration file for the server. You can edit this file, and the files it includes, in a text editor. You can also change the location of the *server.xml* file. However, for most configurations you do not need to do this.

The *bootstrap.properties* file is used to specify properties that need to be available before the main configuration is processed. If you update the *bootstrap.properties* file, you must restart the server for the changes to take effect.

### Example

All the elements that can be configured in the *server.xml* file, and the files it includes, are described in “Liberty features” on page 483. However, the only required element is a server definition:

```
<server/>
```

Beyond this server definition, you only specify overrides and additions to the default configuration values. For example, to change the transaction timeout value, you specify:

```
<transactions timeout="30" />
```

Some attributes can have multiple values. For example, you use a list of values to define the features that are to be provided by the server:

```
<server>
 <featureManager>
 <feature>servlet-3.0</feature>
 <feature>localConnector-1.0</feature>
 </featureManager>
</server>
```

See also “Adding and removing Liberty features” on page 968.

Where multiple instances of a resource type can be configured, for example applications or data sources, you need only provide the attributes that are unique for the resource. You can let the other attributes use the default values, or override them as needed. Therefore the contents of the `server.xml` file can be brief. For example, here is a complete server configuration to run a web application:

```
<server>
 <featureManager>
 <feature>servlet_3.0</feature>
 </featureManager>
 <application name="snoop" location="/mywebapps/snoop" id="snoop" type="war"/>
</server>
```

For detailed information about specific aspects of server configuration, see the subtopics.

## Customizing the Liberty environment

You can customize the Liberty environment by using certain specific variables to support the placement of product binary files and shared resources in read-only file systems.

### About this task

The Liberty specific environment variables in the following list can be configured in the `server.env` file to customize the Liberty environment. The `${wlp.install.dir}` configuration variable has an inferred location that is always set to the parent of the directory that contains the launch script.

- **WLP\_USER\_DIR**

This environment variable can be used to specify an alternative location for `${wlp.user.dir}`. This variable must be an absolute path. If this variable is specified, the runtime environment looks for shared resources and server definitions in the specified directory. The `${server.config.dir}` is equivalent to `${wlp.user.dir}/servers/serverName`. If this environment variable is not specified, `${wlp.user.dir}` is set to `${wlp.install.dir}/usr`.

- **WLP\_OUTPUT\_DIR**

This environment variable can be used to specify an alternative location for server generated output such as logs, the workarea directory, and generated files. Files in the logs directory can include `console.log`, `messages.log`, and any generated FFDC files. Generated files can include server dumps that are created with the **server dump** or **server jvadbump** command. This variable must be an absolute path. If this environment variable is specified, `${server.output.dir}` is set to the equivalent of `WLP_OUTPUT_DIR/serverName`. If this environment variable is not specified, `${server.output.dir}` is the same as `${server.config.dir}`.

When the server command is used, the server process uses the output directory as its current working directory.

- **WLP\_DEBUG\_ADDRESS**

This environment variable can be used to specify an alternative port when you run the server in debug mode. The default value is 7777. When Liberty is run in debug mode from the server command, the following values are set `JAVA_DEBUG="-Dwas.debug.mode=true`  
`-Dcom.ibm.websphere.ras.inject.at.transform=true`

-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=\${WLP\_DEBUG\_ADDRESS}". However, if you run Liberty from the **ws-server.jar** executable JAR file, or the embedded Liberty server SPI, then you must use the same settings to enable debug mode for Liberty.

You can specify WLP\_OUTPUT\_DIR, WLP\_USER\_DIR, and WLP\_DEBUG\_ADDRESS environment variables in server.env files. You can also specify JVM options in jvm.options files. Both server.env and jvm.options files work only when you use the server management script. If you use the **ws-server.jar** executable JAR file to launch your server, these files are not supported.

## Procedure

- Specify environment variables by using server.env files.

You can use server.env files at the installation and server levels to specify environment variables such as JAVA\_HOME, WLP\_USER\_DIR, and WLP\_OUTPUT\_DIR. For example:

```
Use a specific Java binary
JAVA_HOME=/opt/ibm/java-i386-60/jre
JAVA_HOME=c:\Java
```

### Note:

- The server.env files support only key=value pairs.
- Empty lines and lines that start with the # character are ignored.
- There are no escape characters; all characters are literal, including backslashes and leading and trailing white space.
- There should be no white space surrounding the equals "=" sign.
- Shell and variable expansion are not supported.
- WLP\_USER\_DIR can be specified only in the \${wlp.install.dir}/etc/server.env file because the purpose of this variable is to specify where the remaining configuration is located. After the remaining configuration is found and merged, no further configuration in a different location is expected, or supported.

The server management script searches for server.env files in two locations: \${wlp.install.dir}/etc/server.env and \${server.config.dir}/server.env. If both files are present, the contents of the two files are merged; values in the server-level file take precedence over values in the runtime-level file.

You can also specify these environment variables in the shell environment, but the server.env files take precedence over those variables.

- Customize JVM options by using jvm.options files.

You can use jvm.options files at the runtime and server levels to specify more server startup options, for example, **-X** arguments. The options are applied when the start, run, and debug actions are started through the server management script. Be sure to specify only one option per line. For example:

```
Set the maximum heap size to 1024m.
-Xmx1024m

Set a system property.
-Dcom.ibm.example.system.property=ExampleValue

Enable verbose output for class loading.
-verbose:class

Enable verbose garbage collection.
-verbose:gc

Specify an alternate verbose garbage collection log on IBM Java Virtual Machines only.
-Xverbosegclog:verbosegc.log

Specify additional verbose garbage collection options on HotSpot Java Virtual Machines only.
```

```
-Xloggc:verbosegc.log
-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps
-XX:+PrintHeapAtGC
```

The server management script searches for `jvm.options` in two locations: `${wlp.install.dir}/etc/jvm.options` and `${server.config.dir}/jvm.options`. If both files are present, the options in the `${server.config.dir}/jvm.options` file are used.

**Note:**

- Do not put property values in quotes.
- Empty lines and lines that start with the # character are ignored.
- There are no escape characters; all characters are literal, including back-slashes and leading and trailing white space.
- There should be no white space surrounding the equals "=" sign.
- Shell and variable expansion are not supported.

## What to do next

If you start the Liberty server by using the server script, all of the operating system environment variables from the current session are available. If you start the server by using either the MBean or the Admin Center, the only available environment variables are those variables that are available to a remote command that is running on that system.

## Administering Liberty from the command line

You can use the server command and `ws-server.jar` executable JAR file to create a server, to start, or stop a server, to check if it is running, or debug a server.

### About this task

The `wlp/bin` directory contains a script called `server` to help with controlling the server process. The syntax of this script is as follows:

```
server <action> [server] [options]
```

For available values of the **options**, see “server command options” on page 950.

**Avoid trouble:** The administrative console allows a start and a stop of a Liberty server that is a cluster member of an auto-scalable cluster, but only when the server is in maintenance mode. Starting or stopping a Liberty server from the command line when the Liberty server is a cluster member of an auto-scalable cluster can lead to unpredictable results.

This script supports the following actions:

**create** A command that creates a new server.

**run** A command that launches the server in the foreground.

**debug** A command that runs the named server in the console foreground after a debugger connects to the debug port. The default port is 7777. You can use the `WLP_DEBUG_ADDRESS` variable to specify an alternative port.

**dump** A command that creates a snapshot of a server and saves the result into an archive file for further tuning and diagnosis.

#### **javadump**

A command that creates a snapshot of the server Java virtual machine (JVM) and saves the result into files. Each dump type creates a file, but not all dump types are supported by all virtual

machines. The default directory for dump files is `${server.output.dir}`. To set a different default directory, you must use an IBM JVM and set the following environment variables:

- **IBM\_HEAPDUMPPDIR**
- **IBM\_COREDIR**
- **IBM\_JAVACOREDIR**

### **package**

A command that packages a server.

**start** A command that launches the server as a background process.

**stop** A command that stops a running server.

**status** A command that checks to see whether a specified server is running.

### **version**

A command that displays the version information of current server and Java runtime environment.

**help** A command that gets command-line script help, including details of additional options.

**Note:** If a server is not specified on the command line, the action is performed against the default server instance, `defaultServer`, if it exists.

You can also carry out similar actions by using the executable JAR file `ws-server.jar` that is in the `${wlp.install.dir}/bin/tools` directory.

## **Example**

To run the server script on Windows systems:

```
server.bat create server_name
server.bat package server_name
server.bat run server_name
server.bat help server_name
```

To run the server script on other systems:

```
server create server_name
server package server_name
server run server_name
server help server_name
```

To run the executable JAR file `ws-server.jar` without using the server script:

```
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar server_name --create
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar server_name
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar --help
```

The `--help` option provides information about additional command-line parameters for the executable JAR file `ws-server.jar`, such as `--stop`, `--version`, `--clean`, `--include`.

## **server command options**

The **server** command supports starting, stopping, creating, packaging, and dumping a Liberty server. This topic describes all available options and exit codes that you can use with the **server** command and the equivalent executable jar file `ws-server.jar`.

## **Syntax**

The command syntax is as follows:

```
server action serverName [options]
```

where the value of **action** represents the operation that you can perform on a Liberty server. See available administration operations for Liberty from the command prompt.

**Note:** If a server is not specified on the command line, the action is performed against the default server instance, `defaultServer`, if it exists.

## Options

The following options are available for the **server** command:

**--archive**=*"path\_to\_the\_target\_archive\_file"*

Specifies a target file for the *package* or *dump* operation. This path can be either a relative path, which is relative to the installation root directory of Liberty, or an absolute path. The default archive target is a compressed file with the server name, which is stored in the installation root directory. Use quotation marks if the value contains spaces. You can use this option for both *package* and *dump* operations.

### Distributed operating systems

IBM i

If you specify a `.jar` extension for your archive file name, the **server** command creates a new self-extracting archive file from which the Liberty server can be installed by using the **java** command; for more information, see “Installing Liberty by extracting a Java archive file” on page 836. The `.jar` extension facility is not available on the z/OS platform.

**--clean**

Cleans all persistent cached information that is related to the specified server instance, which includes OSGi resolver metadata and persistent OSGi bundle data. If you use this option, the server will be required to recompute any cached data at the next startup, which might take more time than a restart that can reuse cached data.

**Note:** This option is not necessary for normal operation. IBM service might request that you use this option when you provide an interim fix, or if there is a suspected problem with the cached data. This option might also be necessary if you are developing a product extension, and you are either updating OSGi manifests or planning to clear persistent OSGi bundle data.

**--include**=*package\_option*

Specifies the files that you want to package, where *package\_option* can take one of the following values:

- `all` specifies to package all the files in the Liberty installation directory. If the `${WLP_USER_DIR}` and `${WLP_OUTPUT_DIR}` are defined in the `server.env` file, the files under them are packaged. This value applies only to the *package* operation.
- `usr` specifies to package the files in the `${WLP_USER_DIR}` directory. This value applies only to the *package* operation.
- `minify` specifies to package only those parts of the runtime environment, and files in the `${WLP_USER_DIR}` directory, that are required to run the server, which minimizes the size of the resulting archive. This value applies only to the *package* operation.

**--include**=*diagnose\_option,diagnose\_option,...*

Specifies the type of diagnostic information to be captured. The value of **--include** is a comma-delimited list, which can contain any of the following values:

- `heap` is used to help diagnose the excessive memory consumption and memory leaks, which shows live objects in the memory and references between them. On IBM J9 virtual machines, the resulting file is named `heapdump.date.time.processID.sequenceNumber.phd`. On HotSpot virtual machines, the resulting file is named `java.date.time.processID.sequenceNumber.hprof`. This value applies to both the *dump* and *javadump* operations
- `system` is also used to help diagnose the excessive memory consumption and memory leaks, but they are also useful for finding defects in the virtual machine. These dumps are only

supported on IBM J9 virtual machines. The resulting file is named `core.date.time.processID.sequenceNumber.dmp`. This value applies to both the *dump* and *javadump* operations.

- `thread` is used to help diagnose hung threads, deadlocks, and can sometimes be used for diagnose excessive CPU issues. These dumps are always created with the server `javadump` command. On IBM J9 virtual machines, the resulting file is named `javacore.date.time.processID.sequenceNumber.txt`. On HotSpot virtual machines, the resulting file is named `javadump.date.time.processID.sequenceNumber.txt`. This value can also be applied to the *dump* operation.

**Note:** The thread dump type is supported only when the server is running on the Java SDK. If the server is started with a JRE, an error is reported indicating that the server does not support the dump type. This restriction applies to HotSpot virtual machines only; the thread Java dump type is supported on any IBM JVM (JRE or SDK).

`--os=os_value,os_value,...`

Specifies the operating systems that you want the packaged server to support. Supply a comma-separated list. The default value is `any`, indicating that the server is to be deployable to any operating system supported by the source.

To specify that an operating system is not to be supported, prefix it with a minus sign (“-”). For a list of operating system values, refer to the OSGi Alliance web site at the following URL: <http://www.osgi.org/Specifications/Reference#os>.

This option applies only to the *package* operation, and can be used only with the `--include=minify` option. If you exclude an operating system, you cannot later include it if you repeat the *minify* operation on the archive.

## Server Process

The server process is created by using the environment variables that are specified in the `server.env` file. The following JVM options are added by default:

- The `-javaagent:wlp/bin/tools/ws-javaagent.jar` option is required for trace, monitoring, and other server capabilities.
- The `-Xshareclasses` and related options enable the shared class cache on supported IBM J9 virtual machines. The cache directory is set to `WLP_OUTPUT_DIR/.classCache`.
- The `-XX:MaxPermSize` option increases the size of the permanent generation for HotSpot virtual machines prior to Java 8. You can set the `WLP_SKIP_MAXPERMSIZE` environment variable to `true` to avoid this default option, which avoids warnings such as:

```
Java HotSpot(TM) Client VM warning: ignoring option MaxPermSize=256m; support was removed in 8.0
```

You can use the `jvm.options` file to override these default JVM options or add additional JVM options. For more information about the `server.env` and `jvm.options` file, see “Customizing the Liberty environment” on page 947.

**UNIX** **IBM i** **8.5.5.1** By default, the `server` command sets the `umask` value to deny all permissions to “Other” users before running the action. However, you can set the `WLP_SKIP_UMASK` environment variable to `true` to avoid setting the `umask` value.

The current working directory of the server process is set to the server output directory.

**UNIX** **IBM i** The `server` command creates a process ID (PID) file when you start the server and deletes the PID file when you stop the server. By default, the PID file is set to `WLP_OUTPUT_DIR/.pid/serverName.pid`. The absolute path of the PID file can be changed by setting the `PID_FILE` environment variable, or the absolute path of the PID directory can be changed by setting the `PID_DIR` environment variable.



The standard output and error from the server process is output to the foreground console when you use the **run** and **debug** actions and is redirected to the `WLP_OUTPUT_DIR/serverName/logs/console.log` file by default when you use the **start** action. The log name can be changed by setting the `LOG_FILE` environment variable, and the log directory can be changed by setting the `LOG_DIR` environment variable. For more information about the logging configuration, see “Logging and Trace” on page 1461.

The **stop** action prevents new application requests from entering the server, which allows existing requests some time to complete. After that time, the remaining server components are stopped and the server process exits. Application requests that do not complete in the allowed time fail, but their exact behavior depends on their activity when the server components stopped.

## Exit codes

The following exit codes are available for the **server** command and the equivalent executable JAR file **ws-server.jar**:

- 0 OK. 0 indicates successful completion of the requested operation. For server status, 0 indicates that the server is running.
- 1 For server status, 1 indicates that the server is not running. For other operations, it indicates invocation of a redundant operation. For example, starting a started server or stopping a stopped server. This code might also be returned by JVM if invalid Java options are used.
- 2 The server does not exist.
- 3 An unsupported action was called on a running server. For example, the server is running when the package action is called.
- 4 An unsupported action was called on a stopped server. For example, the server is not running when the dump action is called.
- 5 Unknown server status. For example, the `workarea` directory is missing, or the Attach API fails to work.
- >=20 Return codes greater than or equal to 20 indicates that an error occurred while performing the requested . Messages are printed and captured in log files with more information about the error.

## Usage

The following examples demonstrate the correct syntax:

```
server run
server start myserver --clean
server package myserver --archive="archivefile.zip" --include=all
server dump myserver --archive="c:\mybackup\myserver.zip" --include=thread
server javadump myserver
server javadump myserver --include=heap,system
```

## iAdmin command

IBM i

The command supports operating a Liberty server on the IBM i platform. The command file is in the `wlp/lib/native/os400/bin` directory. The command file is a script named `iAdmin`.

## Syntax

The command syntax is as follows:

```
iAdmin task [options]
```

where the value of **task** can be one of the following options:

- **POSTINSTALL**
- **PREUNINSTALL**
- **GRANTAUTH**

The **POSTINSTALL** task configures the server start command to launch servers as jobs in the QWAS85 subsystem. Additionally, the task:

- Configures Liberty to run servers under the QEJBSVR user profile.
- Configures the default JDK location by setting WLP\_DEFAULT\_JAVA\_HOME in file wlp/etc/default.env to the location of the 32 bit version of the minimum supported Java level.
- Adds an entry for the product in the IBM i native product registry.
- Creates IBM i native libraries and objects such as the QWAS85 subsystem and the QEJBSVR user profile.

Call the iAdmin **POSTINSTALL** command only after Installing Liberty by extracting an archive file or when Installing Liberty resources by using the job manager.

The **GRANTAUTH** task grants the QEJBSVR user profile the necessary file permission and ownership for the server role. The **POSTINSTALL** sets the file ownership and authorities correctly for the QEJBSVR user profile. However, if you create files manually, or if you modify the authorities on files used by the Liberty server, you can call the iAdmin **GRANTAUTH** command to ensure that QEJBSVR has the correct authorities.

The **PREUNINSTALL** task removes the native libraries and objects created by the **POSTINSTALL** task. Call the iAdmin **PREUNINSTALL** command before removing the Liberty application-serving environment from your system, but you only need do so if Liberty was installed by executing a JAR file.

**Note:**

- You must have \*ALLOBJ and \*SECADM special authority to use the **POSTINSTALL** and **PREUNINSTALL** commands.
- You must have \*ALLOBJ special permission, own, or have \*OBJMGT authority to all objects in the specified directory subtrees to use the **GRANTAUTH** command.
- After running the **POSTINSTALL** task, you must also have \*ALLOBJ and \*SECADM special authority to start and stop the Liberty server.

**Options**

The following options are available for the **iAdmin** command:

**--outputdir wlp\_user\_dir**

The directory for server generated files. This option must only be an absolute path, is optional for the **GRANTAUTH** task, and is ignored for all other tasks. When not specified, the default location for server generated output is used.

**--rolename role\_name**

The role that the user profile is assigned. The server role is the only currently supported role. This option is required for the **GRANTAUTH** task and is ignored for all other tasks.

**--userdir wlp\_user\_dir**

The directory containing shared resources and server definitions. This option can only be an absolute path, is optional for the **GRANTAUTH** task, and is ignored for all other tasks. When not specified, the default location for shared resources and server definitions is used.

**--userprofilename user\_profile\_name**

The user profile to grant authority to. **QEJBSVR** is the only currently supported user profile name for the server role. This option is required for the **GRANTAUTH** task and is ignored for all other tasks.

## Usage scenarios

The following examples demonstrate correct syntax. Run the command in any of the following examples on one line.

- Configuring Liberty to start as a job in the QWAS85 subsystem and to swap to the QEJBSVR user profile when running.  
wlp/lib/native/os400/bin/iAdmin POSTINSTALL
- Granting the server role to the QEJBSVR user profile for the shared resources, server definitions, and output locations configured for this Liberty runtime environment.  
wlp/lib/native/os400/bin/iAdmin GRANTAUTH --rolename server --userprofilename QEJBSVR
- Removing the native libraries and objects created by the **POSTINSTALL** task.  
wlp/lib/native/os400/bin/iAdmin PREUNINSTALL

## application client commands



8.5.5.6

The **client** command supports create, run, debug, package and help actions.

### Syntax

The wlp/bin directory contains scripts that are called **client** and **client.bat** to help with running client applications. The syntax of these scripts are as follows:

```
client <action> client_name [options]
```

where the value of **action** represents the operation that you can perform on an application client.

**Attention:** If an application client name is not specified on the command line, the task is taken against the default application client instance, defaultClient.

### Actions

The **client** script supports the following actions:

**create** A command that creates a new application client.

**run** A command that launches the application client in the foreground.

**debug** A command that runs the named application client in the console foreground after a debugger connects to the debug port. The default port is 7778.

**package** A command that packages an application client.

**help** A command that gets command-line script help, including details of more options.

### Options

The **client** script supports the following options:

**--archive="path to the target archive file"**

Specify the archive target to be generated by the package action. You can specify the target as an absolute path or as a relative path. If this option is omitted, the archive file is created in the client output directory. The target file name extension might influence the format of the generated archive. The default archive format for the package action is pax on z/OS and zip on all other platforms. The jar archive format produces a self-extracting jar file that similar to the original installer archive.

**--clean**

Clean all cached information related to this client instance.

**--include=value,value,...**

A comma-delimited list of values. The valid values vary depending on the action.

**template="templateName"**

Specify the name of the template to use when creating a new client.

**--autoAcceptSigner**

Automatically accept and store the signer certificate in the client truststore without being prompted to examine the server certificate.

**Example**

The following examples show the application client command actions that you can run on Windows systems: Windows

```
client.bat create client_name
client.bat run client_name
client.bat help
```

The following examples show the application client command actions that you can run on other systems:

```
client create client_name
client run client_name
client help
```

**Running an application client from the command line:**  8.5.5.6

You can use the client run task to run an application client.

**Before you begin**

Before you can run an application client, you must create a client and add a configuration for your client application in the `client.xml` file. See the [Creating a Liberty client manually](#) topic for examples of how you would create the application client.

**About this task**

The `wlp/bin` directory contains scripts that are called `client` and `client.bat` to run client applications. The syntax of these scripts is as follows:

```
client action <client_name> [options]
```

**Procedure**

Use the following command to run the client:

```
client run client_name
```

Windows

```
client.bat run client_name
```

where `client_name` is the name of the client.

**Attention:** If a default client exists, it runs `defaultClient`. If a default client does not exist, `defaultClient` is created, then runs (and likely fails because there is no application that is configured).

## configUtility command

8.5.5.5

The **configUtility** command enables you to download configuration snippets from the IBM WebSphere Liberty Repository. The command also enables you to replace configuration snippet variables with your input values. The Liberty repository *config snippets* are samples of Liberty server configurations for specific tasks.

For information about accessing and using the Liberty repository, see “Liberty Repository” on page 573.

### Syntax

The command syntax is as follows:

```
configUtility action | configUtility configSnippet [options]
```

where *action* can take one of the following values:

**find** Display a list of all configuration snippets in the repository.

**8.5.5.6** Specify a string to filter the list to configuration snippets that have the specified string in the description. Enclose strings that contain spaces in double-quotation marks.

**help** Display help information for a specified action.

**install** Download the configuration snippet from the repository or use a local configuration snippet for variable substitution.

The *configSnippet* variable is the name of the configuration snippet. Run the **configUtility find** command to get the names of configuration snippets in the repository.

### Options

The following options are available for the **configUtility install configSnippet** command:

**--info** List all variable options in the configuration snippet. Return an empty list if the configuration snippet has no variables for substitution.

**--v[variable]=value**

Replace configuration snippet variables found by the **--info** option with your input values. The utility identifies these variable using **--v[variable]**. Do not include the brackets ([]) in the command.

**--createConfigFile=path**

Optional. The utility writes the code snippet to the file specified by *path* instead of to the console screen. Add the provided code snippet to the `server.xml` configuration to include the specified file.

**--encoding=[xor|aes]**

Optional. Specify the keystore password encoding. Supported encodings are xor and aes. The default encoding is xor.

**--key=key**

Optional. Specify a key to be used when encoding using AES. This string is hashed to produce an encryption key which is used to encrypt and decrypt the password. Optionally, provide the key to the server by defining the variable `wlp.password.encryption.key` whose value is the key. If this option is not provided, a default key is used.

**--useLocalFile=file**

Use a configuration snippet from a local file system. You must specify the file path. This option replaces specifying a configuration snippet name.

## Usage

The following examples demonstrate correct syntax:

```
configUtility find
configUtility find filter_string
configUtility find "filter_string"
configUtility help
configUtility install configSnippet
configUtility install configSnippet --info
configUtility install configSnippet --vvariable=value
configUtility install configSnippet --createConfigFile=C:/wlp/usr/servers/server1/snippet-include.xml
configUtility install configSnippet --encoding=aes --key=myAESkey
configUtility install --useLocalFile=C:/wlp_temp/mySnippet.xml
```

**Tip:** If your option value has spaces, you must enclose it in double quotation marks ("). For example, if the file path for the `--createConfigFile` option is `C:\Program Files\mySnippets\snippet-include.xml`, specify `--createConfigFile="C:/Program Files/mySnippets/snippet-include.xml"` in the command.

### CAUTION:

**Different operating system might treat some characters differently. For the Windows environment, if you have ! in your input string, it needs to be escaped by the ^ character. For example,**

```
D:\Liberty\images\855\Liberty855\wlp\bin>configUtility createConfigFile="a^!"
```

### Downloading and customizing configuration snippets from the command line: 8.5.5.5

You can use the **configUtility** command to download configuration snippets from the IBM WebSphere Liberty Repository. You can also use the command to replace configuration snippet variables with your input values. The Liberty repository *config snippets* are samples of Liberty server configurations for specific tasks.

### Procedure

1. Open a command line, then change directory to the `wlp/bin` directory.
2. Generate a list of configuration snippets in the Liberty repository.

Run the following command to write a list of configuration snippets to your console screen:

```
configUtility find
```

From this generated list, you can determine the name of a configuration snippet to download and, optionally, modify.

8.5.5.6 You can specify a string to filter the list to configuration snippets that have the specified string in the description. For example, for a list of security-related configuration snippets, run:

```
configUtility find security
```

Enclose strings that contain spaces in double-quotation marks; for example:

```
configUtility find "ejb security"
```

3. Optional: Generate a list of all variable options for a configuration snippet.

Using the name of configuration snippet obtained in step 2, run the following command to write a list of variable options for the snippet to your console screen. For *configSnippet*, specify the name of the snippet:

```
configUtility install configSnippet --info
```

The command returns an empty list if the configuration snippet has no variables for substitution.

4. Download a configuration snippet.

Using the name of the configuration snippet obtained in step 2, run the following command to write the configuration snippet to your console screen. For *configSnippet*, specify the name of the snippet:

```
configUtility install configSnippet
```

To write the configuration snippet to a file instead of to your console screen, add the `--createConfigFile` option and specify a file path. You can include the generated file in a `server.xml` configuration file using an include statement. For example, to write the configuration snippet to the `C:\wlp\usr\servers\server1\snippet-include.xml` file on your local file system, run the following command:

```
configUtility install configSnippet --createConfigFile=C:/wlp/usr/servers/server1/snippet-include.xml
```

**Tip:** If the file path has spaces, you must enclose it in double quotation marks ("). For example, if the file path is `C:\Program Files\mySnippets\snippet-include.xml`, specify `--createConfigFile="C:/Program Files/mySnippets/snippet-include.xml"` in the command.

To download and replace configuration snippet variables found by the `--info` option with your input values, use the `--v[variable]=value` option:

```
configUtility install configSnippet --v[variable]=value
```

For example, to specify a value of `user1` for the `adminUser` variable in the `remoteAdministration` snippet, run:

```
configUtility install remoteAdministration --vadminUser=user1
```

This command writes a configuration for Remote Administration to a console screen, with `user1` replacing the `adminUser` variable:

```
<quickStartSecurity userName="user1" userPassword="{adminPassword}"/>
```

For more information about command options, see “`configUtility` command” on page 957.

## Running the ddlGen utility

8.5.5.6

You can generate data definition language (DDL) if there are features in the server configuration that require access to a database.

### Before you begin

You must start the server before running the `ddlGen` utility.

### About this task

The utility generates data definition language (DDL) for each feature that is configured in the server that requires access to a database. You can change the path that the `ddlGen` utility uses to search for the server by exporting the environment variable, `WLP_USER_DIR`, in the command line where the utility is run.

### Procedure

1. In the `server.xml` file, add the `localConnector-1.0` feature under the `featureManager` tag.

```
<featureManager>
 <feature>localConnector-1.0</feature>
</featureManager>
```
2. In a command line, run the `wlp/bin/ddlGen {generate|help} <server_name>` command, where `<server_name>` is the name of the server that you want to generate DDL for.

### Results

The following table shows the nonzero codes that might be returned:

Table 77. Return codes and explanations for the `ddlGen` utility

Return Code	Explanation
0	Success. The DDL is generated to <code>{server.output.dir}/ddl</code> .

Table 77. Return codes and explanations for the ddlGen utility (continued)

Return Code	Explanation
20	The action provided is not valid.
21	The server was not found. Message CWWKD0100E shows the file system directory where the utility looked for the server. This location can be changed by exporting the variable, WLP_USER_DIR, in the command line where the utility is run.
22	The localConnector feature is not present in the server configuration or the server was not started.
23	The MBean that generates DDL was not found.
24	The MBean that generates DDL reported an error. The server logs contain more details about the error.
255	An unexpected error occurred.

## Generating Liberty configurations schema from the command line

Use the ws-schemagen.jar tool that is in the installation's bin/tools directory to generate the schema for Liberty core and other installed products extensions in a single output file.

### Syntax

The command syntax is as follows:

```
java [JVM options] -jar ws-schemagen.jar [options] outputFile
```

### Options

The following options are available:

#### --encoding=charset

Where *charset* is the character set to use when you are creating the output file.

#### --ignorePidsFile=fileName

Where *fileName* is the name of the file name that contains a list of pids to ignore.

#### --locale=language

Where *language* specifies the language to use when you are creating the output file. This string consists of the ISO-639 two-letter lowercase language code, optionally followed by and underscores and the ISO-3166 uppercase two-letter country code.

### Usage examples

The following example generates the installed products' schema and stores in a file called schema.xml:

```
java -jar ws-schemagen.jar schema.xml
```

The following example generates the installed products' schema and stores in a file that is called schema.xml using the Brazilian Portuguese language:

```
java -jar ws-schemagen.jar schema.xml --locale=pt_BR
```

The following example displays help information:

```
java -jar ws-schemagen.jar --help
```

## Generating a Liberty server dump from the command line

From the command line, you can use the server dump or server javadump command to capture status information for a Liberty server.



## About this task

The server dump command is useful for problem diagnosis of a Liberty server because the result file contains server configuration, log information, and details of the deployed applications in the workarea directory. The command can be applied to either a running or a stopped server.

For a running server, the following information is also included:

- State of each OSGi bundle in the server
- Wiring information for each OSGi bundle in the server
- Component list that is managed by the Service Component Runtime (SCR) environment
- Detailed information of each component from SCR
- Configuration administration data of each OSGi bundle
- Information about registered OSGi services
- Runtime environment settings such as Java virtual machine (JVM), heap size, operating system, thread information, and network status

| The server javadump command is useful for diagnosing problems at the JVM level, such as hung threads,  
| deadlocks, excessive processing, excessive memory consumption, memory leaks, and defects in the virtual  
| machine. The command can be used only on a running server. Each dump type creates a file, but not all  
| dump types are supported by all virtual machines. See “server command options” on page 950. The  
| default directory for dump files is `${server.output.dir}`. To set a different default directory, you must  
| use an IBM JVM and set the following environment variables:

- | • **IBM\_HEAPDUMPDIR**
- | • **IBM\_COREDIR**
- | • **IBM\_JAVACOREDIR**

## Procedure

1. Open a command line, then change directory to the wlp/bin directory.
2. Capture the status information by using one of the following command-line tools. If you do not specify a server name, defaultServer is used.

- To create a snapshot of the server status, use server dump command.

```
– Distributed operating systems IBM i
server dump server_name --archive=package_file_name.dump.zip --include=heap
```

| where *package\_file\_name*.dump.zip is a file name that you choose. This file name can include a  
| full path name. If the full path is omitted, a compressed file called *package\_file\_name*.dump.zip  
| is created in default directory `${server.output.dir}`.

The --include parameter is optional. You can request additional memory dump types. For example, --include=heap option requests a heap dump; --include=thread,heap,system option requests a thread dump, a heap dump, and a system dump.

- To create a snapshot of the JVM status, use server javadump command.

```
– Distributed operating systems IBM i
server javadump server_name --include=heap
```

The --include parameter is optional. You can request additional memory dump types. For example, --include=heap option requests a heap dump; --include=heap,system option requests a heap dump and a system dump. The output files are created in the default directory `${server.output.dir}`. To set a different default directory, you must use an IBM JVM and set the **IBM\_HEAPDUMPDIR**, **IBM\_COREDIR**, and **IBM\_JAVACOREDIR** environment variables.

**Note:** The resulting file is created by using UTF-8 encoding for entry names, so the tool that you use to open the file must be able to use UTF-8 encoding for entry names. The `jar` command in a Java SDK uses this format.

## Results

If the specified server does not exist, the command does not succeed. If the specified server exists, a result file is created that contains the status information of the server.

## Packaging a Liberty server from the command line

From the command line, you can create a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server. You can also choose to exclude the runtime binary files from the compressed file.

## About this task

The Liberty server is lightweight, and therefore you can easily package a server installation in a compressed file. You can store this package, distribute it to colleagues, and then use it to deploy the installation to a different location or to another machine, or even embed the installation in a product distribution.

The server installation that you want to package cannot already be joined to a collective. You can only package a stand-alone server.

**Note:** **Distributed operating systems** The resulting file is created by using UTF-8 encoding for entry names, so the tool that you use to open the file must be able to use UTF-8 encoding for entry names. The `jar` command in a Java SDK uses this format.

## Procedure

To package a Liberty server from the command line, complete the following steps:

1. Open a command line, then change directory to the `wlp/bin` directory.
2. Stop the server.
3. Run the **package** command to create a package.

**8.5.5.5** You can package the Liberty profile server or the runtime.

- Package the Liberty server.

The default archive format is `.zip` on all platforms apart from z/OS where it is `.pax`. You can also generate a `.jar` archive.

If you do not specify a server name, `defaultServer` is used. If you do not specify the **--archive** parameter, the value of `server_name` is used for `package_file_name`, and the compressed file is created in the `${server.output.dir}` directory.

Choose the correct command for your environment.

– **Distributed operating systems** **IBM i**

Use this command to generate a `.zip` archive.

```
server package server_name --archive=package_file_name.zip --include=all
```

where `package_file_name.zip` is a file name that you choose. This file name can include a full path name. If the full path is omitted, a compressed file called `package_file_name.zip` is created in the `${server.output.dir}` directory.

– **Distributed operating systems** **IBM i**

Use this command to generate a `.jar` archive. The advantage of a `.jar` archive is that the scripts in the `bin` directory keep their permissions, so they are executable when the package is installed.

```
server package server_name --archive=package_file_name.jar --include=all
```

where *package\_file\_name.jar* is a file name that you choose.

For more information about extraction options with this archive file, see “Java archive file extraction options” on page 837.

You can also use the **--include** option with this command. For example, the **--include=all** option packages the runtime binaries and the relevant files in the `${WLP_USER_DIR}` directory; the **--include=usr** option packages only relevant files in the `${WLP_USER_DIR}` directory, effectively excluding the runtime binaries from the compressed file.

The **--include=usr** option is not valid with an archive format of `.jar`.

If you use the **--include=minify** option, the **server** command packages only those parts of the runtime environment, and files in the `${WLP_USER_DIR}` directory, that are required to run the server. This option significantly reduces the size of the resulting archive.

The parts of the runtime environment that are retained by the **minify** operation depend on the features that are configured in the server that you are packaging. Only those features that are required to run the server are retained, and the remaining features are removed. Therefore, you cannot later enable a feature that has been removed. For example, if only the `servlet-3.0` feature is retained, you cannot later enable the `jpa-2.0` feature.

You can repeat the **minify** operation to further reduce the size of the archive if the configuration is changed. There is, however, no reverse operation for the **minify** operation, so if you later require one or more features that have been removed, you must begin again with a complete Liberty server.

While the **minify** operation is running, the server is temporarily started, and you see the associated messages. For this reason, you cannot use the **--include=minify** option with a server that is not able to be started, but you can package it with the **--include=all** or **--include=usr** options.

You can specify the operating systems that you want the packaged server to support by using the **--os** option with the **--include=minify** option.

For example, to package a server with z/OS support removed, use the following command:

```
server package --archive="nozoz.zip" --include=minify --os=-z/OS
```

To package a server with OS/400 support retained, but z/OS support removed, use the following command:

```
server package --archive="small.zip" --include=minify --os=OS/400,-z/OS
```

To package a server that supports only Linux, use the following command:

```
server package --archive="linux.zip" --include=minify --os=Linux
```

- **8.5.5.5** Package the Liberty runtime.

Create a runtime archive that contains the `wlp` directory, but does not contain the `usr` directory. The naming convention for a server package is *package\_name.zip*; for example, `CustomerPortalApp.zip`. To create a runtime archive, run the **package** command without a server name and with the **--include=wlp** option:

```
server package --include=wlp
```

To specify a package file name and target location, add the **--archive=package\_path\_name** option; for example:

```
server package --include=wlp --archive=c:\temp\myPackage.zip
```

If you do not specify a valid package name or target location with the **--archive** option, then the command creates the `wlp.zip` runtime archive in the `$WLP_OUTPUT_DIR` location, which is the `${wlp.install.dir}/usr/servers` directory by default. The target location must exist before running the command. Thus, if the target location is `c:\temp`, the `C:\temp` directory must exist and have write permission for the command to write the archive to the `C:\temp` directory.

## Running a Liberty server from a JAR file

8.5.5.9

You can start a Liberty server from a Java archive (JAR) file. This method provides a compact, portable way of starting a Liberty server. You create the JAR file by using the Liberty server command and then run as an executable JAR file by using the `java -jar` command.

## Create the runnable JAR file

You can specify `minify` to get the smallest archive possible. You must specify a JAR type archive to get a runnable JAR file. The default archive type is `.zip` on all platforms except `z/OS`, where the only supported type is `pax`. For example:

```
server package <server name> --include=[minify,]runnable --archive=<jar file name>.jar
```

## Run the JAR file

Run the JAR file by using the standard `java` command with the `-jar` option, for example:

```
java -jar <jar file name>.jar
```

## Operation

When the JAR file runs, it gets extracted to a temporary location and then the server runs in the foreground, started by the Liberty server run command. All output is written to `stdout` or `stderr`. By default, files are extracted to temporary locations:

- For Windows: `%HOMEPATH%/wlpExtract/<jar file name>_nnnnnnnnnnnnnnnnnn`
- For all other platforms: `$HOME/wlpExtract/<jar file name>_nnnnnnnnnnnnnnnnnn`

You can control the output location by using the `WLP_JAR_EXTRACT_ROOT` or `WLP_JAR_EXTRACT_DIR` environment variable.

## Stopping the server

To stop the Liberty server press **Ctrl-C**. When the Liberty server stops, the extraction directory is automatically deleted. If you stop the active shell in any other way, the extraction directory is not cleaned up automatically, you must clean it up manually.

## Run in debug mode

You can run the Liberty server in debug mode if you set the environment variable `WLP_JAR_DEBUG` before you start the server.

## Controlling output

By default, server output is written to the extraction directory, which is deleted when the server stops. If you want to save the output, specify a durable output location by using the `WLP_OUTPUT_DIR` environment variable before you start the server.

## Two-phase commit transactions

By default, two-phase commit transactions are disabled because the transaction logs are in the expansion directory and are deleted when the Liberty server is stopped. Therefore, transaction recovery is not possible.

To enable two-phase commit, configure the transaction log to be in a durable location in the file system or an RDBMS and set the `WLP_JAR_ENABLE_2PC` environment variable.

To configure the transaction log, use either the `transactionLogDirectory` or `dataSourceRef` attribute on the transaction element in your `server.xml` configuration.

## Running under CYGWIN

Running a Liberty server JAR file in a CYGWIN shell has two requirements:

1. Specify the `WLP_JAR_CYGWIN` environment variable.

This variable causes the Liberty server JAR runner to do UNIX-style file and process handling when it is running in the CYGWIN environment.

2. Run under the bash shell, not mintty.

Automatic extraction file deletion occurs only when you run under the bash shell. You can run under mintty, but you must delete the extraction files manually. Mintty does not forward the necessary signal that is required to trigger Java shutdown hooks.

## Environment variable reference

Table 78. The environment variable names and their definitions

Environment variable name	Description
<code>WLP_JAR_EXTRACT_ROOT</code>	Extracts JAR file to directory <code>\${WLP_JAR_EXTRACT_ROOT}/&lt;jar file name&gt;_nnnnnnnnnnnnnnnnnnnn</code>
<code>WLP_JAR_EXTRACT_DIR</code>	Extracts JAR file to directory <code>\${WLP_JAR_EXTRACT_DIR}</code> .
<code>WLP_OUTPUT_DIR</code>	Writes Liberty server output files to directory <code>\${WLP_OUTPUT_DIR}</code> .
<code>WLP_JAR_DEBUG</code>	Runs Liberty server by using server debug <code>&lt;server name&gt;</code> instead of server run <code>&lt;server name&gt;</code> .
<code>WLP_JAR_ENABLE_2PC</code>	Set to value true to enable 2PC when the runnable JAR file runs.
<code>WLP_JAR_CYGWIN</code>	Set to value true if you are running the JAR file under CYGWIN.

## Starting and stopping a server from the command line

You can use the server tasks to start or stop a server.

### About this task

The `wlp/bin` directory contains a script called `server` to help with controlling the server process. The syntax of this script is as follows:

```
server action serverName [options]
```

For available values of `[options]`, see “server command options” on page 950.

**Note:** If a server is not specified on the command line, the action is performed against the default server instance, `defaultServer`, if it exists.

### Procedure

- Use the following command to start the server:

```
server start serverName
```

where `serverName` is the name of the server.

- Use the following command to stop the server:

```
server stop serverName
```

where `serverName` is the name of the server.

**Note:** Normal server stop includes a quiesce stage before the server is shutdown. The quiesce stage, a period of 30 seconds, allows for services to perform pre-shutdown work, for example, stopping inbound listeners while allowing existing requests to complete. Applying the `--force` option to the stop command will skip the quiesce stage. The `--force` option will have no effect if **server stop** was already invoked. If you use the `--force` option, you may see unexpected exceptions in the `messages.log` file that occur after the **server stop** command was received by the server.

## Example

To start or stop a server by using the server script on Windows systems:

```
server.bat start serverName
server.bat stop serverName
```

To start or stop a server by using the server script on other systems:

```
server start serverName
server stop serverName
```

## Starting and stopping a Liberty server as a Windows service

### Windows

You can optionally register the Liberty server as a Microsoft Windows Service program when you run on Windows operating systems. You can manually start, stop, and unregister the Liberty server as a Windows service. You can also use the Microsoft Windows Services program to change Liberty to start automatically as a Windows service when the Windows server starts.

## Before you begin

Set the `JAVA_HOME` environment variable before you run the following commands. You can use the `server.env` file path to set the `JAVA_HOME` value reliably. For more information about using the `server.env` file, see “Customizing the Liberty environment” on page 947.

## About this task

The commands to register, start, stop, and unregister Liberty as a Windows service use the same `server.bat` file that is used when you start the Liberty server from the command line. For available values of `[options]`, see “server command options” on page 950.

## Procedure

- Use the following command to register the server:

```
server registerWinService serverName
```

where `serverName` is the name of the server.

The Windows service name and Windows service display name is the `serverName`. After you register, you can find and administer the Liberty server as a service by using **regedit** from the Windows Registry program. Entries for the service in the registry are in the following locations:

```
HKEY_LOCAL_MACHINE->SOFTWARE->Wow6432Node->Apache Software Foundation->Procrun 2.0->serverName
HKEY_LOCAL_MACHINE->SYSTEM->CurrentControlSet->services->serverName
```

- Use the following command to start the server:

```
server startWinService serverName [options]
```

where `serverName` is the name of the server.

- Use the following command to stop the server:

```
server stopWinService serverName
```

where *serverName* is the name of the server.

- Use the following command to unregister the server:

```
server unregisterWinService serverName
```

where *serverName* is the name of the server.

## What to do next

When you install a fix pack by using archives, the directory under which the fix pack is installed can change from the directory where the Liberty server was running before you installed the fix pack. If you are running a Liberty server as a Windows service, then unregister that Liberty server as a Windows service before you install the fix pack. After you install the fix pack, register the Liberty server as a Windows service again.

If you need to manually remove Liberty as a Windows service, then update the Windows registry by using the Windows **regedit** program. The two entries that need to be removed from the Windows Registry are in the following locations:

```
HKEY_LOCAL_MACHINE->SOFTWARE->Wow6432Node->Apache Software Foundation->Procrun 2.0->serverName
HKEY_LOCAL_MACHINE->SYSTEM->CurrentControlSet->services->serverName
```

## Using Ant to automate tasks for Liberty

Apache Ant is a Java library tool for automating the build process. You can use Ant tasks that are provided by Liberty to manage the server and applications.

### Before you begin

**Important:** The `wlp-anttasks.jar` file that is included with Liberty is planned to be removed. For more information, see Removal notices. An open source Ant plug-in with more tasks for Liberty is available. The more recent Ant plug-in has a different antlib namespace, `xmlns:wlp="antlib:net.wasdev.wlp.ant"`. For more information about the open source Ant plug-in for Liberty, see the GitHub repository.

The Liberty Ant plug-in `wlp-anttasks.jar` files are located in Maven Central. If you want to use these tasks in your build script, you must make sure the plug-in is available on the Ant class path. Copy the plug-in file `wlp-anttasks.jar` to the `/lib` directory of the Ant installation, and declare the `antlib` namespace in the `build.xml` file. See the following example:

```
<project xmlns:wlp="antlib:net.wasdev.wlp.ant">
 ...
</project>
```

The namespace can be any string, provided you avoid name conflicts. After that, you must use the namespace as a prefix of the Ant tasks for Liberty. For example, you must use `wlp:server` when calling the server task.

### About this task

You can create build scripts that use these Ant tasks to package, install, and test your application on Liberty.

## Using an OSGi console

Eclipse Equinox currently provides an OSGi console that you can use to aid with debugging. This console is not available by default. You can enable this console in the OSGi framework that is running within Liberty by using the `osgiConsole-1.0` feature and by specifying a port to attach to.

## About this task

Liberty uses the Eclipse Equinox implementation of the OSGi core specification. Equinox currently provides an OSGi console. To enable this console, you first allocate a specific port to it by setting the **osgi.console** property in the `bootstrap.properties` file. Then you can use Telnet to connect to the console on that port, and explore the OSGi framework.

### Procedure

- Add the `osgiConsole-1.0` Liberty feature to your `server.xml` file.  

```
<feature>osgiConsole-1.0</feature>
```
- Allocate a specific port to the OSGi console.  
You set the OSGi console port by specifying the **osgi.console** property. You set this property as a bootstrap property in the `bootstrap.properties` file. See “Specifying Liberty bootstrap properties” on page 897.  

```
osgi.console=5471
```

The OSGi console is disabled when the **osgi.console** property is not set.
- Use Telnet to connect to the OSGi console port.  

```
telnet localhost 5471
```
- Use the console to explore the framework.  
The available commands vary, depending on the OSGi framework being used. Command-line help provides enough information to get started.

## Adding and removing Liberty features

Features are the units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server. To add or remove a Liberty feature, you add or remove an XML snippet in the `<feature>` subelement of the `server.xml` configuration file. When you add or remove Liberty features, the changes are applied dynamically.

### Distributed operating systems

## Before you begin

You can add and remove Liberty features as described in this topic, or as described in “Editing the Liberty configuration by using developer tools” on page 938.

## About this task

For a list of the main Liberty features, including the XML snippets that enable them, see “Liberty features” on page 483.

### Procedure

To add or remove Liberty features, complete the following steps:

1. Open the `server.xml` configuration file for editing.  
Where *path\_to\_liberty* is the location you installed Liberty on your operating system, and *server\_name* is the name of your server.  
You can do this using a text editor. By default, the path and file name for the configuration root document file is `path_to_liberty/wlp/usr/servers/server_name/server.xml`. However, you can change the path. See “Customizing the Liberty environment” on page 947.
2. Add or remove features in the configuration file.  
The set of features is enclosed within the `<featureManager>` element, and each feature within the `<feature>` subelement. For example:



```

<server>
 <featureManager>
 <feature>servlet-3.0</feature>
 <feature>localConnector-1.0</feature>
 </featureManager>
</server>

```

The matching of feature names is not case-sensitive; the following example is also a valid server configuration:

```

<featureManager>
 <feature>Servlet-3.0</feature>
 <feature>localConnector-1.0</feature>
</featureManager>

```

3. Save the changes to the configuration file.

## Results

Your changes are applied. If the server is running, the changes are applied dynamically.

## Using include elements, variables, and Ref tags in configuration files

You can keep all your configuration settings in a single `server.xml` file, or you can use include elements to consolidate configuration settings from separate files. You can use variables in the configuration to avoid hardcoding values that might not be appropriate when the configuration is reused in different environments. You can use Ref tags to refer to (and thereby reuse) existing code blocks that are defined elsewhere in the configuration.

### Procedure

- “Using include elements in configuration files”
- “Using variables in configuration files” on page 970
- “Using Ref tags in configuration files” on page 972

### Using include elements in configuration files

You can keep all your configuration in a single `server.xml` file, or you can use include elements to consolidate configurations from separate files to create the structure that is most useful to you.

### About this task

It can be easier to maintain a complex configuration by splitting it across a set of files. For example:

- You might want to include a file that contains variables that are specific to the local host, so that your main configuration can be used on multiple hosts.
- You might want to keep all of the configuration for a particular application in a separate file that can be versioned with the application itself.

### Example

This is the syntax for including a configuration file. You can set the **optional** attribute as `true` if you want to skip the include file when it cannot be found :

```

<include optional="true" location="pathname/filename"/>
or
<include optional="true" location="url"/>

```

The following list shows the possible locations; they are searched in the order shown.

1. in a location specified relative to the parent file
2. in the server configuration directory

3. in a location specified as an absolute path
4. on a web server

To ensure that your include configuration behaves predictably, you need to be aware of the following processing rules for included configuration files:

- For singleton services such as logging, or application monitoring, entries are processed in the order they appear in the file and later entries add to or override previous ones. This is also true for configuration instances, for example an application or data source, where the configuration instances have the same ID.
- Include statements can be placed anywhere within the `<server />` element.
- Each included file must contain a `<server />` element that matches the one in the parent configuration file.
- Included files can nest other included files.
- Each included file is logically merged into the main configuration at the position that the `<include />` statement occurs in the parent file.

In the following example, the primary server configuration file `server.xml` includes the contents of the `blogDS.xml` configuration file, which is located in the shared configuration directory. The `blogDS.xml` file contains a data source definition. This definition has been put in a separate configuration file so that it can be included in several different `server.xml` files, and thereby used across multiple server instances.

Here is example code from the `server.xml` file:

```
<server>
 <featureManager>
 <feature>servlet-3.0</feature>
 <feature>jdbc-4.0</feature>
 </featureManager>
 <application id="blog" location="blog.war" name="blog" type="war"/>
 <include optional="true" location="{shared.config.dir}/blogDS.xml"/>
</server>
```

Here is the example code from the `blogDS.xml` file:

```
<server>
 <dataSource id="blogDS" jndiName="jdbc/blogDS" jdbcDriverRef="derbyEmbedded">
 <properties createDatabase="create" databaseName="C:/liberty/basics/derby/data/blogDB" />
 </dataSource>
 <jdbcDriver id="derbyEmbedded"/>
 <library>
 <fileset dir="C:/liberty/basics/derby" includes="derby.jar" />
 </library>
</jdbcDriver>
</server>
```

## Using variables in configuration files

You can use variables in the configuration to avoid hardcoding values that might not be appropriate when the configuration is reused in different environments.

### About this task

Variables can be defined by setting a property in any of the following places:

- in the server configuration file, or an included file
- in the `bootstrap.properties` file

The following predefined variables can be referenced:

- directory properties, see “Directory locations and properties” on page 894

- JVM system properties
- process environment variables

If the same variable is specified in multiple places, the precedence is as follows:

- variables in `bootstrap.properties` override the process environment variables
- variables in `server.xml`, or included XML files, override the variables in `bootstrap.properties` and process environment variables

**Best practice:** Variables that are specific to a particular server, for example port numbers, are specified in the `bootstrap.properties` file, allowing the `server.xml` to be shared across multiple servers while you keep those values different in each server. Variables that are shared across a group of servers, for example database configuration for a particular host, is better specified in an xml file that is included into the parent configuration file.

**Best practice:** Variable names must begin with an alphabetic character, and must contain the following characters only: alphabetic characters, numeric characters, and the "\_" and "." characters.

## Procedure

- Specify a variable in a configuration file.

The variable definition syntax is `variable_name=value`. If the value contains a path, it is normalized during configuration processing by replacing repeated forward and backward slashes with a single forward slash, unless the value starts with double forward or backward slashes, which remain unchanged.

**Best practice:** If you need to set the value of a variable to contain repeated forward slashes, as are sometimes used for JDBC driver connection URLs, break the value into two parts at the double slashes. By placing the double forward slashes as the initial characters, normalization is avoided. For example, to store the value `"jdbc:db2://host_name.com"`, use two variables:

```
URL_PART_1="jdbc:db2:"
URL_PART_2="//host_name.com"
```

Variables that are defined in the configuration files are scoped to the configuration elements by which they are used. For example, the following code fragment creates a variable that is called `updateTrigger_var` to be used in **applicationMonitor** configuration elements:

```
<applicationMonitor updateTrigger_var="mbean" />
```

To create a variable that is used in a particular configuration instance (such as an application or resource entry), you must also specify the instance identifier. For example: `<httpEndpoint id="defaultHttpEndpoint" HTTP_default_var="8889" />`

- Specify a variable in the `bootstrap.properties` file.

Variables that are defined in the `bootstrap.properties` file are not scoped to particular configuration elements. You enter the variables as key-value pairs. For example:

```
HTTP_default_var=8006
```

- Use a defined variable in the configuration.

The variable substitution syntax is `${variable_name}`. Multiple variable values may be concatenated by specifying `${variable_name1}${variable_name2}`. For example, to use the `HTTP_default_var` variable, add the following code fragment to the configuration file:

```
<httpEndpoint id="defaultHttpEndpoint"
httpPort="${HTTP_default_var}">
</httpEndpoint>
```

- Use variable element in the configuration

You can use the `variable` element to define a variable globally in the server configuration. If the same variable is defined in an included file, it is overridden by the one in the `server.xml` file. For example, to use the `variable` element, add the following code fragment to the configuration file:

```
<variable name="HTTP_default_var" value="8889" />
```

- Use process environment variables in the configuration

Process environment variables are available if you use the env. configuration variable prefix, for example:

```
<fileset dir="${env.LIBRARY_DIR}" includes="*.jar"/>
```

For more information about specifying environment variables, see “Customizing the Liberty environment” on page 947.

- **8.5.5.4** Use variable expressions in configuration

For configuration variables, you can use a limited variable expression syntax with the format `${<operand><operator><operand>}`. The description of the variable is as follows:

#### operand

Operands can either be long integer literals or the name of a variable that contains a long integer value. Variable names must begin with an alphabetic character, and must contain the following characters only: alphabetic characters, numeric characters, and the “\_” and “.” characters.

#### operator

The available operators are as follows:

- + for addition
- - for subtraction
- \* for multiplication
- / for division

If the expression cannot be parsed, a non-integer value is used, or an arithmetic error occurs, then the behavior is undefined.

For example, if the `HTTP_port_base` variable is defined, a variable expression might be used to define multiple `httpEndpoints`:

```
<httpEndpoint id="defaultHttpEndpoint" httpPort="${HTTP_port_base+0}"/>
<httpEndpoint id="httpEndpoint2" httpPort="${HTTP_port_base+1}"/>
```

- Override inheritable attributes in the configuration

You can override the default values of inheritable attributes in the configuration. The inheritable attributes are listed on the **\*\*\*\* MISSING FILE \*\*\*\*** page with an `Inherits` type. For example, the **onError** attribute is one of inheritable attributes. You can define a variable name for the **onError** attribute globally by either setting it in the `bootstrap.properties` or `server.xml` file with a `variable` element. If the same variable name is specified in both files, the value in the `server.xml` file is used. If the attribute is not explicitly set in either of two files, it uses the default value. If an invalid value is set to the inheritable attribute, the attribute value falls back to the global value defined in `bootstrap.properties` or `server.xml` file or the default value if not defined at the global level.

Another example is logging properties in Liberty. See “Logging and Trace” on page 1461.

## Using Ref tags in configuration files

You can define a common configuration element, then reuse that definition by referring to it (using a `Ref` tag) from elsewhere in the configuration. `Ref` tags can be used in the same configuration file that contains the element definition, or in an included configuration file.

### About this task

Different approaches are used to specify relationships between the required configuration elements. For example, the following data source definitions are all valid. The first uses no `Ref` tags, the second uses a combination of direct element definition and `Ref` tags, and the third uses `Ref` tags only.

## Example

Example 1: Using no Ref tags.

```
<dataSource id="blogDS" jndiName="jdbc/blogDS">
 <properties createDatabase="create" databaseName="C:/liberty/basics/derby/data/blogDB"/>
 <jdbcDriver>
 <library>
 <fileset dir="C:/liberty/basics/derby" includes="derby.jar"/>
 </library>
 </jdbcDriver>
 <connectionManager maxPoolSize="10"/>
</dataSource>
```

Example 2: Combining direct element definition and Ref tags.

```
<dataSource id="blogDS" jndiName="jdbc/blogDS" connectionManagerRef="derbyPool">
 <properties createDatabase="create" databaseName="C:/liberty/basics/derby/data/blogDB"/>
 <jdbcDriver libraryRef="derbyLib"/>
</dataSource>

<connectionManager id="derbyPool" maxPoolSize="10"/>

<library id="derbyLib"/>
 <fileset dir="C:/liberty/basics/derby" includes="derby.jar"/>
</library>
```

Example 3: Using Ref tags only (except for the properties element, which is only permitted as nested).

```
<dataSource id="blogDS" jndiName="jdbc/blogDS"
 connectionManagerRef="derbyPool" jdbcDriverRef="derbyEmbedded">
 <properties createDatabase="create" databaseName="C:/liberty/basics/derby/data/blogDB"/>
</dataSource>

<connectionManager id="derbyPool" maxPoolSize="10"/>

<jdbcDriver id="derbyEmbedded" libraryRef="derbyLib"/>

<library id="derbyLib" filesetRef="derbyFileset"/>

<fileset id="derbyFileset" dir="C:/liberty/basics/derby" includes="derby.jar"/>
```

## Using the configuration dropins folder to specify server configuration

8.5.5.5

You can specify additional configuration files in the configDropins directory without specifying include elements in the server.xml file.

### Procedure

1. Create a configDropins directory under the `usr/servers/server_name` directory.

- `usr/servers/server_name/configDropins/overrides`

If you want to add configuration files to replace anything in the server.xml file of the server, create a configDropins/overrides directory. For example, to change ports that are defined in the server.xml, use a configDropins/overrides directory.

- `usr/servers/server_name/configDropins/defaults`

If you want the server.xml file to be the master configuration, but want to specify defaults for elements that the server.xml does not define, create a configDropins/defaults directory. For example, if you want developers to be able to provide configuration, but you want the server.xml to be the master configuration and you do not want the server.xml changed, use a configDropins/defaults directory.

2. Place the server configuration files in either the configDropins/overrides or configDropins/defaults directory.

Both directories are monitored for updates so that when you add, remove, or update configuration files, the runtime configuration is updated dynamically.

If there are any conflicts, the following rules determine precedence:

- The configuration that is specified in the configDropins/overrides directory takes precedence over the configuration in the server.xml file. Configuration specified in server.xml file takes precedence over configuration that is specified in the configDropins/defaults directory.
- Configuration from files in both the configDropins/defaults and configDropins/overrides directories take precedence over any default configuration that is specified by a feature.
- The configuration files in the dropins directory are processed in alphabetical order. A later configuration overrides an earlier one. As an example, if configDropins/defaults contains a.xml, b.xml and c.xml, the configuration from c.xml takes precedence over b.xml, and b.xml takes precedence over a.xml.

**Note:** To maintain consistency across platforms, file names are converted to lower case before sorting alphabetically. This means that if two files are specified in the same dropins directory that have the same name except for case variations (such as extraConfig.xml and ExtraConfig.xml), the ordering behavior is indeterminate.

3. Optional: Turn off configuration monitoring. See “Controlling dynamic updates” on page 975.

## ID variables that refer to configuration files

8.5.5.6

The Liberty run time occasionally needs to refer to a configuration element from the server.xml file. This action can occur in several ways, such as in the text of a message or in a file name.

The Liberty run time uses an xpath-style syntax to refer to configuration elements. The element type is printed first, followed by brackets that contain the ID of the configuration element. If the configuration element is nested inside another configuration element, the inner configuration element is preceded with a forward slash separating the inner and outer elements.

- For example, the following databaseStore configuration element is referred to as databaseStore[DBTaskStore] because the databaseStore is not nested and has an ID value of DBTaskStore.

```
<server>
 <databaseStore id="DBTaskStore">
 ...
 </databaseStore>
</server>
```

- The following data source configuration element is referred to as databaseStore[DBTaskStore]/dataSource[DataSource0] because the data source is nested underneath databaseStore, databaseStore has an ID value of DBTaskStore, and data source has an ID value of DataSource0.

```
<server>
 <databaseStore id="DBTaskStore">
 <dataSource id="DataSource0">
 ...
 </dataSource>
 </databaseStore>
</server>
```

- In some cases, a configuration element does not have an ID. In this case, an ID is generated. For example, the following data source configuration element might be referred to as databaseStore[default-0]/dataSource[DataSource0] because the databaseStore does not define an ID.

```
<server>
 <databaseStore>
 <dataSource id="DataSource0">
```

```
...
</dataSource>
</databaseStore>
</server>
```

## Controlling dynamic updates

There are three types of dynamic update that can be controlled through configuration: changing the server configuration; adding and removing applications; updating installed applications. For all deployed applications, you can configure whether application monitoring is enabled and how often to check for updates to applications. For the “dropins” directory, you can also configure the name and location of the directory and choose whether to deploy the applications that are in the directory.

### About this task

By default, deployed applications are monitored for updates, and the updates are dynamically applied to the running application. This applies both to applications that are deployed through configuration entries, and those deployed from the “dropins” directory. You can change these default behaviors by setting the **config** and **applicationMonitor** elements in the `server.xml` configuration file. You can use a text editor to do this, or you can use the developer tools and select **Configuration Admin Service** or **Application Monitor** in the server configuration design view.

See also the descriptions of the **config** and **applicationMonitor** elements in \*\*\*\* MISSING FILE \*\*\*\*.

The default settings for application monitoring are as follows:

```
<applicationMonitor updateTrigger="polled" pollingRate="500ms"
 dropins="dropins" dropinsEnabled="true"/>
```

The default settings for configuration monitoring are as follows:

```
<config updateTrigger="polled" monitorInterval="500ms"/>
```

#### Notes:

- The **updateTrigger** property has three possible values:

**polled** The runtime environment scans the `server.xml` file for changes using the timing interval specified by the **monitorInterval** property.

#### **mbean**

The runtime environment only looks for updates when prompted to do so through a call to an MBean. This is the mode that is used by the developer tools to update the `server.xml` file, unless you override it.

#### **disabled**

The updates are not dynamically applied.

- When you specify the **pollingRate** property or the **monitorInterval** property, you include the unit of time after the number:
  - ms (milliseconds)
  - s (seconds)
  - m (minutes)
  - h (hours)
- The **dropins** property specifies the name of the directory used as the “dropins” directory.
- The **dropinsEnabled** property is a boolean property that determines whether the applications in the “dropins” directory are deployed.

### Procedure

- Configure dynamic changes to the server configuration.

Changes to the `server.xml` file, or any files it includes, are detected by the runtime environment and applied to the active configuration. You can disable this behavior by setting the `config` element in the `server.xml` file:

```
<config updateTrigger="disabled"/>
```

You can also control dynamic updates to the server configuration through a provided mbean by setting the `config` element in the `server.xml` file:

```
<config updateTrigger="mbean"/>
```

You can then use the `FileNotificationMbean` to notify the server which configuration file or files you want to be dynamically reprocessed.

- Configure dynamic addition and removal of applications.

As described in Chapter 9, “Deploying applications in Liberty,” on page 1329, applications can be dynamically added to and removed from the server runtime environment through two mechanisms:

- By adding or removing application entries in the `server.xml` file.

If you disable dynamic changes to the server configuration as described in the previous step, then adding or removing application entries has no effect on a running server. Your changes are only applied at the next server restart. The changes are picked up immediately, if you update application entries using the developer tools.

- By moving application files into and out of the “dropins” directory.

This behavior can be controlled by setting the `applicationMonitor` element in the `server.xml` file. For example, to disable dynamic installation of applications from the “dropins” location, create an entry as follows:

```
<applicationMonitor dropinsEnabled="false"/>
```

- Configure dynamic updates to installed applications.

By default, if you add, remove or modify any files within a deployed application, or you replace the whole application with an updated version, the previous version is automatically stopped and the new version is started. This process applies for any deployed application, whether the application is in the “dropins” directory or at a location defined in the `server.xml` file. You can control this behavior by setting the `applicationMonitor` element in the `server.xml` file. For example, to disable dynamic update of all applications, create an entry as follows:

```
<applicationMonitor updateTrigger="disabled"/>
```

- Configure the name and location of the “dropins” directory.

By default, the “dropins” directory is `${server.config.dir}/dropins`. You can change this by setting the `applicationMonitor` element in the `server.xml` file. For the location, you can use any known variable, or a property in the `bootstrap.properties` file, or an absolute path, or a path relative to the server directory. For example, both the following settings point to the same location:

```
<applicationMonitor dropins="${server.config.dir}/applications" />
<applicationMonitor dropins="applications" />
```

**Restriction:** For web service applications, if the service client and service provider are not in the same application and the WSDL file in the service provider application is changed, you need to restart the web service client application manually to avoid the WSDL definition cache issue.

## Configuring class loaders and libraries for Java EE applications

By default, each application can access a set of provided APIs and its own internal classes and libraries. You can override the default settings, and configure class loading for each application.

### About this task

Each Java EE application has its own class loader in a running Liberty server. Liberty assumes some default settings for all Java EE applications, so that they can access the supported specification APIs (for example the servlet APIs if the servlet feature is enabled), and the IBM APIs. By default, each application



can access these provided APIs and access its own internal classes and libraries. If you have to override the default settings and configure class loading for your application, complete one or more of the following tasks.

**Note:** If you use configuration to override the default settings, you cannot deploy the application by dropping it into the “dropins” directory.

## Procedure

- “Using a Java library with a Java EE application”
- “Sharing a library across multiple Java EE applications”
- “Accessing third-party APIs from a Java EE application” on page 979
- “Removing access to third-party APIs for a Java EE application” on page 980
- “Overriding a provided API with an alternative version” on page 980
- “Providing global libraries for all Java EE applications” on page 978

## Using a Java library with a Java EE application

One way of using Java libraries with an application is to include them in the application itself. This might not always be desirable or appropriate, especially if the application is already packaged and does not include the library.

### About this task

In the following example, a library called Alexandria consists of two files:

- alexandria-scrolls.jar and
- commons-lang.jar

An application called Scholar, running on a server called Academy, needs access to this library.

## Procedure

1. Create a mylib/Alexandria directory in the servers/Academy directory under the `${WLP_USER_DIR}` directory.

For example: `wlp/usr/servers/Academy/mylib/Alexandria`.

2. Copy the alexandria-scrolls.jar and commons-lang.jar files into the new folder.
3. Configure class loading for the application, so that the Alexandria library is loaded.

In the server.xml file, or an included file, add the following code:

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader>
 <privateLibrary>
 <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
 </privateLibrary>
 </classloader>
</application>
```

**Note:** The `<privateLibrary>` element can also take a `filesetRef` attribute with a comma-separated list of `<fileset>` element IDs.

## Sharing a library across multiple Java EE applications

Libraries can be shared across multiple Java EE applications. All the applications can use the same classes at run time, or each application can use its own separate copy of those classes loaded from the same location.

## About this task

In the following example, a library called Alexandria consists of two files:

- alexandria-scrolls.jar and
- commons-lang.jar

An application called Scholar and an application called Student are running on a server called Academy, and both need access to this library.

## Procedure

1. Create a mylib/Alexandria directory in the servers/Academy directory under the `${WLP_USER_DIR}` directory.

For example: `wlp/usr/servers/Academy/mylib/Alexandria`.

2. Copy the alexandria-scrolls.jar and commons-lang.jar files into the new folder.
3. Configure class loading for the application, so that the Alexandria library is loaded.

In the server.xml file, or an included file, define the library by adding the following code:

```
<library id="Alexandria">
 <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

**Note:** The `<library>` element can also take a `filesetRef` attribute with a comma-separated list of `<fileset>` element IDs.

4. Reference the library from the applications, so that both these applications share a single copy of the library.

In the server.xml file, or an included file, add the following code:

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader commonLibraryRef="Alexandria" />
</application>

<application id="student" name="Student" type="ear" location="student.ear">
 <classloader commonLibraryRef="Alexandria" />
</application>
```

**Note:** The `<commonLibraryRef>` element can take a comma-separated list of library IDs.

5. Optional: Configure another application to have its own set of classes loaded from the same JAR files.

For example, if another application called Spy needs its own copy of the classes, the same physical files on disk can be used. In the server.xml file, or an included file, add the following code:

```
<application id="spy" name="Spy" type="war" location="spy.war">
 <classloader privateLibraryRef="Alexandria" />
</application>
```

**Note:** The `<privateLibraryRef>` element can take a comma-separated list of library IDs.

## Providing global libraries for all Java EE applications

You can provide global libraries that can be used by any Java EE application. You do this by putting the JAR files for those libraries in a global library directory, then specifying use of global libraries in the class loader configuration for each application. However, the global libraries cannot be used by other applications, for example, by OSGi applications.

## About this task

Under the user directory specified by using the environment variable `WLP_USER_DIR`, there are the following locations in which you can place global libraries:

- `${shared.config.dir}/lib/global`

- `${server.config.dir}/lib/global`

If there are files present in these locations at the time an application is started, and that application does not have a `<classloader>` element configured, the application uses these libraries. If a class loader configuration is present, these libraries are not used unless the global library is explicitly referenced.

**Attention:** If you use global libraries, you are advised also to configure a `<classloader>` element for every application. The servlet specification requires applications to share the global library class loader in their class loader parent chain. This breaks the separation of class loaders for each application that is otherwise possible. So, applications are more likely to have long-lasting effects on classes loaded in Liberty and on each other, and class space consistency issues are more likely to arise between applications, especially as features are added and removed from a running server. None of these considerations apply for applications that specify a `<classloader>` element in their configuration, because they maintain this separation.

## Example

In the following example, an application called `Scholar` is configured to use a common library called `Alexandria`, and also to use the global library.

In the `server.xml` file, or an included file, enable the global library for an application by adding the following code:

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
 <classloader apiTypeVisibility="spec" commonLibraryRef="Alexandria, global" />
</application>
```

The settings for the global library can also be configured explicitly, as a library element with the special ID `global`. For example:

```
<library id="global">
 <fileset dir="/path/to/folder" includes="*.jar" />
</library>
```

## Accessing third-party APIs from a Java EE application

By default, Java EE applications do not have access to the third-party APIs available in Liberty. To enable this access, the application must be configured in the `server.xml` file, or an included file.

### About this task

In the following example, an application that is called `Scholar` needs access to the third-party APIs that are available in Liberty.

The application also uses a common library called `Alexandria`. This library is located in the `${server.config.dir}/mylib/Alexandria` directory.

**Avoid trouble:** Third party APIs might not remain compatible after an upgrade. For more information, see “Liberty externals support” on page 14.

### Procedure

1. Configure class loading for the application, so that the application can access the third-party APIs.

The default value for the `apiTypeVisibility` attribute of the `classloader` element is `spec,ibm-api,api`. Where `spec` represents public specification APIs available for both compile and run time, `ibm-api` represents APIs available in Liberty, and `api` represents public APIs available for both compile and run time. Including `third-party` in the `apiTypeVisibility` attribute of the `classloader` element makes third party APIs available.

In the `server.xml` file, or an included file, configure the API type visibility by adding the following code:

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader apiTypeVisibility="spec, ibm-api, third-party" commonLibraryRef="Alexandria" />
</application>
```

2. Optional: If the application uses any common libraries, set those libraries to use the same API type visibility setting.

In the `server.xml` file, or an included file, add the following code:

```
<library id="Alexandria" apiTypeVisibility="spec, ibm-api, third-party">
 <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

## Removing access to third-party APIs for a Java EE application

By default, Java EE applications do not have access to the third-party APIs available in Liberty. You can also remove access explicitly in the `server.xml` file, or an included file.

### About this task

In the following example, an application called `Scholar` has previously been configured to access third-party APIs, as described in “Accessing third-party APIs from a Java EE application” on page 979. You want to remove this access, and to make it explicit in the configuration that the application now uses the default access setting.

The application also uses a common library called `Alexandria`. This library is in the `${server.config.dir}/mylib/Alexandria` directory.

### Procedure

1. Configure class loading for the application, to show that the application can no longer access the third-party APIs.

In the `server.xml` file, or an included file, remove `third-party` from the set of values included for the `apiTypeVisibility` attribute:

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader apiTypeVisibility="spec, ibm-api" commonLibraryRef="Alexandria" />
</application>
```

2. Optional: If the application uses any common libraries, set those libraries to use the same API type visibility setting.

In the `server.xml` file, or an included file, add the following code:

```
<library id="Alexandria" apiTypeVisibility="spec, ibm-api">
 <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

## Overriding a provided API with an alternative version

If an application provides (or uses a library that provides) classes that are also available in Liberty, by default the classes from Liberty are used. To change this so that the application uses the alternative versions of these classes, the application must be configured in the `server.xml` file, or an included file.

### About this task

If a web application includes classes that are also present in the server runtime environment, you might want to control which copy of each of those classes is used by the application. For example, if different versions of the classes are present in both the application and the server runtime environment, you must ensure that the version packaged in the application is used.

By default, classes from the Liberty runtime environment are used by all Java EE applications. You can override this behavior by using the class loader configuration **delegation** attribute. This configuration is specific to a particular application, or to a shared library that can be selected for use by an application.

## Example

In the following example, an application called Scholar needs to use classes that it provides (or that are provided in a library that it uses), rather than using the copies of the classes that are available in Liberty.

- When the classes are packaged within the application, override the default `parentFirst` delegation behavior with a `classloader` element in the `server.xml` configuration file or a file that it includes:

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
 <classloader delegation="parentLast" />
</application>
```

This tells the application class loader to look at the Liberty classes only after looking in the application and its associated libraries for a class.

- When the classes are packaged in a shared library, add the **delegation** attribute to the `classloader` element that configures the use of the shared library as follows:

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
 <classloader delegation="parentLast" commonLibraryRef="mySharedLib"/>
</application>
```

```
<library id="mySharedLib">
 <fileset dir="${server.config.dir}/myLib" includes="*.jar" />
</library>
```

You can also use the **privateLibraryRef** attribute for private libraries in an application. See “Sharing a library across multiple Java EE applications” on page 977.

## Configuring libraries for OSGi applications

8.5.5.6

Each OSGi application can access a set of provided APIs and its own internal classes. Shared libraries can also be configured to provide access to extra packages from shared libraries.

### About this task

Each OSGi application has its own set of OSGi bundles in a running Liberty server. Each OSGi bundle specifies the packages that it needs and the packages it provides for use by other OSGi bundles. Bundles within an OSGi application can access any packages that are provided by other bundles within the same OSGi application. Additionally, OSGi bundles within an OSGi application can access API packages provided by the Liberty server. Shared libraries can also be used to provide API packages for use by OSGi applications.

Libraries can be shared across multiple OSGi applications. All the applications, including Java EE applications, can use the same classes at runtime that are provided by shared libraries.

### Procedure

1. Create a `mylib/osgi` directory in the `servers/defaultServer` directory under `${WLP_USER_DIR}` directory. For example: `wlp/usr/servers/defaultServer/mylib/osgi`.
2. Copy the `osgi-lib.jar` and `commons-lang.jar` files into the new folder.
3. Configure the shared library for the application so that the library is loaded. In the `server.xml` file, or an included file, define the library by adding the following code:

```
<library id="mylib">
 <fileset dir="${server.config.dir}/mylib/osgi" includes="*.jar" scanInterval="5s">
</library>
```

**Note:** The library element can also take a `filesetRef` attribute with a comma-separated list of fileset element IDs.

4. Reference the library as an OSGi library so that OSGi applications can access the packages provided by the library and share a single copy of the library. In the `server.xml` file, or an included file, add the following code:

```
<osgiLibrary libraryRef="myLib"/>
```

5. Optional: Configure the list of packages to make them available for access from OSGi applications. Packages that are contained in the shared library can be accessed by OSGi applications when the library is configured by using the `osgiLibrary` element. Packages can also be listed to give more control over what packages are accessible by OSGi applications. The package syntax uses the OSGi Export-Package header syntax to define each package. To list the packages in the `server.xml` file, or an included file, add the following code:

```
<osgiLibrary libraryRef="myLib">
 <package>org.example.osgi.lib.pkg1; version=1.0</package>
 <package>org.example.osgi.lib.pkg2; version=1.1</package>
</osgiLibrary>
```

**Note:** When no package elements are used, the library is scanned to find the packages that the library provides. Each package that is discovered gets the default version of 0.0.0.

## Configuring JPA for Liberty

8.5.5.6

### About this task

Java Persistence API (JPA) 2.0 for WebSphere Application Server is built on the Apache OpenJPA 2.2.x open source project.

Apache OpenJPA is a compliant implementation of the JPA 1.0 and 2.0 specifications. Using OpenJPA as a base implementation, WebSphere Application Server employs extensions to provide more features and utilities for WebSphere Application Server customers. Because JPA for WebSphere Application Server is built from OpenJPA, all OpenJPA function, extensions, and configurations are unaffected by the WebSphere Application Server extensions. You do not need to make changes to OpenJPA applications to use these applications in WebSphere Application Server.

Java Persistence API (JPA) 2.1 for WebSphere Application Server is built on the EclipseLink open source project. EclipseLink is the reference implementation for all version of the JPA specification. The provider of JPA for this product is `org.eclipse.persistence.jpa.PersistenceProvider`.

### Configuring JPA logging

8.5.5.6

Logging supports viewing, tracing, and troubleshooting the runtime behavior of an application. Each of the JPA features provides different levels of logging for you to specify how detailed you want the logging to be.

### About this task

When using either the `jpa-2.0` or `jpa-2.1` features, you can configure logging to aid in troubleshooting. Become familiar with the logging capabilities of these two features.

#### **jpa-2.0**

There are many supported `jpa-2.0` trace specifications that can be configured through the Liberty configuration. These trace strings can be used in conjunction with any other trace specifications.

#### **Container-managed JPA applications**

- `JPA=all`

Enables all JPA container trace and all OpenJPA tracing

- `openjpa=all`  
Enables all OpenJPA tracing
- OpenJPA specific log channels

`openjpa.jdbc.SQL=all`

```
<server>
...
<logging traceSpecification="openjpa.jdbc.SQL=all"
 traceFileName="trace.log"
 maxFileSize="20"
 maxFiles="10"
 traceFormat="BASIC" />
</server>
```

### Application managed JPA applications

When running a JPA application that is application managed, logging and tracing is controlled by the OpenJPA runtime. All JPA tracing and logging must be configured through OpenJPA persistence properties.

```
<persistence version="2.0">
<persistence-unit>
 <properties>
 <property name="openjpa.Log" value="openjpa.jdbc.SQL=trace"/>
 </properties>
</persistence-unit>
</persistence>
```

### Notable OpenJPA logging persistence properties

`openjpa.ConnectionFactoryProperties=PrintParameters=true`-- If true, SQL bind parameters are included in exceptions and logs.

## jpa-2.1

When the `jpa-2.1` feature is enabled, all JPA logging and tracing is routed through the Liberty loggers.

### Supported trace strings

- `JPA=all`  
Enables JPA container trace and all EclipseLink categories
- `eclipselink=all`  
Enables all EclipseLink trace
- EclipseLink specific log categories
  - `sql`, `transaction`, `event`, `connection`, `query`, `cache`, `propagation`, `sequencing`, `ejb`, `dms`, `metadata`, `weaver`, `properties`, `server`
  - ie: `eclipselink.sql=All` -- Enables EclipseLink SQL trace

```
<server>
...
<logging traceSpecification="eclipselink.sql=all"
 traceFileName="trace.log"
 maxFileSize="20"
 maxFiles="10"
 traceFormat="BASIC" />
</server>
```

### Notable EclipseLink logging persistence properties

`eclipselink.logging.parameters` -- If true, SQL bind parameters are included in exceptions and logs.

## Procedure

In the persistence unit definition in the persistence.xml file, specify the logging level depending on the desired level of logging details that you want. Specify the `eclipselink.logging.level` property where the value is the logging level. For the list of logging levels available, refer to the EclipseLink logging wiki page. The following example will turn on all logging that is available.

```
<persistence-unit name="pu">
 <properties>
 <property name="eclipselink.logging.level" value="ALL"/>
 ...
 </properties>
</persistence-unit>
```

## Configuring the JPA 2.1 schema generator

8.5.5.6

In the `jpa-2.0` feature, which is built on OpenJPA, you can generate data definition language (DDL) or interact directly with the database to define table schemas based on the JPA entity definition by using the SchemaMapper tool. In the `jpa-2.1` feature, which is built on EclipseLink, you can use the new Schema Generator feature added to the JPA 2.1 specification, which has similar functions to the OpenJPA SchemaMapper.

### About this task

If you need functions similar to the OpenJPA SchemaMapper, you can configure the Schema Generator feature that is in the JPA 2.1 specification.

## Procedure

1. In the persistence unit definition, within the `persistence.xml` file, specify the database action property with the possible values of: `none`, `create`, `drop`, `drop-and-create`. Each value corresponds to the action that is taken against the database. The following example causes the tables that correspond to the entities specified in the persistence unit to be dropped and new tables are created in their place.

```
<persistence-unit name="pu">
 <properties>
 <property name="javax.persistence.schema-generation.database.action"
 value="drop-and-create" />
 ...
 </properties>
</persistence-unit>
```

2. Specify the script action property with the possible values of: `none`, `create`, `drop`, `drop-and-create`. If any value other than `none` is specified, you must specify a target property as well. This means that if the script action is `create`, which generates the create statements for the entity definition, you must specify a corresponding create target property with a target file where the statements are written to.

```
<persistence-unit name="pu">
 <properties>
 <property name="javax.persistence.schema-generation.scripts.action"
 value="drop-and-create" />
 <property name="javax.persistence.schema-generation.scripts.create-target"
 value="createTargetFile.ddl"/>
 <property name="javax.persistence.schema-generation.scripts.drop-target"
 value="sampleDrop.ddl"/>
 ...
 </properties>
</persistence-unit>
```

## Disabling the EclipseLink shared object cache

8.5.5.6



The EclipseLink shared object cache contains a subset of all objects that are read and persisted for the persistence unit. The EclipseLink shared cache differs from the local EntityManager/L1/persistence context cache. The shared cache exists during the persistence unit and is shared by all EntityManagers and users of the persistence unit.

## About this task

If you are migrating an existing application or running an environment where your application spans multiple Java Virtual Machines (JVMs), you can disable the EclipseLink shared object cache.

Choose one of the following ways to disable the EclipseLink shared object cache.

## Procedure

- Set the `<shared-cache-mode>NONE</shared-cache-mode>` property in the `persistence.xml` file.

```
<persistence-unit name="pu">
 <shared-cache-mode>NONE</shared-cache-mode>
 <properties>
 ...
 </properties>
</persistence-unit>
```

- Set the `eclipseLink.cache.shared.default` property to `false` in the persistence unit definition that is found in the `persistence.xml` file.

```
<persistence-unit name="pu">
 <properties>
 <property name="eclipseLink.cache.shared.default" value="false" />
 ...
 </properties>
</persistence-unit>
```

## Configuring a web server plug-in for Liberty

You can configure a web server plug-in to receive an HTTP request for dynamic resources. You can forward the request to the Liberty server, which provides high-availability and workload balancing through the web server plug-in.

## Before you begin

1. Install a supported web server, such as the IBM HTTP Server that is included with IBM WebSphere Application Server. See [Installing IBM HTTP server](#) for more information. The web server that is provided with IBM i is already installed with product 5761-DG1 for IBM i V6R1, or 5770-DG1 for IBM i V7R1. The IBM i web server is referred to as the IBM HTTP Server for IBM i. The HTTP Server that is provided with WebSphere Application Server does not run on IBM i and is included for z/OS, V7 and V8.0. To download the V8.5 HTTP server, see [IBM Ported Tools for z/OS](#).
2. Install the web server plug-ins and the WebSphere Customization Toolbox (WCT). To install the web server plug-in, see [Installing and configuring web server plug-ins](#).

**IBM i** For IBM i, see [Installing and configuring web server plug-ins](#). To install the WCT, see [Installing and using the WebSphere Customization Toolbox](#).

**IBM i** For IBM i and z/OS, install the WCT on your workstation. You do not need to install any of the WCT tools. The Java SDK installed with the WCT is used to run the JConsole Java utility in a later step.

## About this task

A web server plug-in is used to forward HTTP requests from a supported web server to one or more application servers. The plug-in checks the request against configuration data in the `plugin-cfg.xml` file. The configuration data maps the URI for the HTTP request to the host name of an application server. The web server plug-in then uses this information to forward the request to the application server.

The procedure applies to Liberty servers not in a collective.

## Procedure

1. Configure the web server plug-in for your chosen web server by using the WCT.
  - When prompted in WCT, choose the **remote** scenario and specify the host name that Liberty is accessible on.
  - Do not copy or run the generated `configureWebserver` script. This script is not required with Liberty.

2. **IBM i** Configure your HTTP server to use the `plugin-cfg.xml` file.

Find the location of your current `plugin-cfg.xml` by finding the value that is specified for the `WebSpherePluginConfig` directive at the end of the configuration file of the HTTP server. For example, `<IHS_ROOT>/conf/httpd.conf`.

Enable the plug-in within the `httpd.conf` file of the web server by using the `LoadModule` phrase, and specify the location of `plugin-cfg.xml` file by using the `WebSpherePluginConfig` phrase. For example:

- On Windows systems: **Windows**

```
LoadModule was_ap22_module "path/to/mod_was_ap22_http.dll"
WebSpherePluginConfig "C:\Program Files\IBM\HTTPServer\conf\plugin-cfg.xml"
```

- On other distributed systems: **AIX** **Linux** **UNIX** **HP-UX** **Solaris**

```
LoadModule was_ap22_module "path/to/mod_was_ap22_http.so"
WebSpherePluginConfig "/opt/IBM/HTTPServer/conf/plugin-cfg.xml"
```

**IBM i** For IBM i and z/OS, see *Configuring IBM HTTP Server* for instructions about enabling the plug-in within the `httpd.conf` file.

3. Optional: If you want the web server plug-in to forward HTTP requests to more than one Liberty server, repeat the previous steps for each additional server. Make sure that you consolidate all the plug-in configurations into one `plugin-cfg.xml` file.
  - You can use a `pluginCfgMerge` utility in the traditional server to merge multiple `plugin-cfg.xml` files. See *Configuring simple load balancing across multiple application server profiles*.
  - You can use a `Plugin Merge` command-line tool available on GitHub to merge individual configuration files into a single `merged-plugin-cfg.xml` file.

## Generating the plugin-cfg.xml file

This information can be used to generate the `plugin-cfg.xml` file for the web server, which is used to forward HTTP requests from a supported web server to one or more application servers. The plug-in takes a request and checks the request against configuration data in the `plugin-cfg.xml` file.

## Before you begin

If an application programmatically modifies the session cookie configuration by using Servlet 3.0 APIs, then the application must be initialized before you generate the `plugin-cfg.xml` file. Otherwise, the **AffinityCookie** attribute that is defined for that application might be wrong. To avoid this problem, you can set **deferServletLoad** to false, start the server, generate the plug-in, and then remove the **deferServletLoad** attribute.

## About this task

A web server plug-in is used to forward HTTP requests from a supported web server to one or more application servers. The plug-in takes a request and checks the request against configuration data in the `plugin-cfg.xml` file. The configuration data maps the URI for the HTTP request to the host name of an application server. The web server plug-in then uses this information to forward the request to the application server.

## Procedure

1. Start the server that hosts your applications, and ensure that the `localConnector-1.0` feature for IBM i and z/OS platforms, or the `restConnector-1.0` feature if you are configuring a plug-in for IBM i or z/OS, and any other required features are included in the server configuration.

In the `pluginConfiguration` element of the server configuration file, you can specify the `webserverPort` and `webserverSecurePort` attributes to forward requests from the web server. By default, the value of `webserverPort` is 80 and the value of `webserverSecurePort` is 443. However, you might want to change these settings. For example, for Linux and similar platforms, if you are a non-root user, you must use port numbers greater than 1024.

For all configurable attributes of the `pluginConfiguration` element, see “Java Servlets 3.1” on page 545.

Here is an example of a `server.xml` server configuration file:

```
<server description="new server">
 <featureManager>
 <feature>localConnector-1.0</feature>
 <feature>jsp-2.2</feature>
 </featureManager>

 <keyStore id="defaultKeyStore" password="{xor}PGY6bW4w0yw+" />

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080">
 <tcpOptions soReuseAddr="true" />
 </httpEndpoint>

 <pluginConfiguration webserverPort="80"
 webserverSecurePort="443"
 sslKeyringLocation="path/to/sslkeyring"
 sslStashfileLocation="path/to/stashfile"
 sslCertlabel="definedbyuser"/>

 <application type="war" id="myapp" name="myapp" location="\${server.config.dir}/apps/myapp.war" />
 <application type="war" id="snoop" name="snoop" location="\${server.config.dir}/apps/snoop.war" />
</server>
```

**IBM i** For IBM i and z/OS, include the `restConnector-1.0` feature instead of the `localConnector-1.0` feature. For details, see [Configuring secure JMX connection to Liberty](#).

**IBM i** Here is an example of a `server.xml` server configuration file for IBM i and z/OS:

```
<server description="new server">
 <!-- Enable features -->
 <featureManager>
 <feature>jsp-2.2</feature>
 <feature>restConnector-1.0</feature>
 </featureManager>

 <keyStore id="defaultKeyStore" password="{xor}PGY6bW4w0yw+" />

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080"
 httpsPort="9443">
 <tcpOptions soReuseAddr="true" />
 </httpEndpoint>

 <quickStartSecurity userName="testuser"
 userPassword="security" />

 <pluginConfiguration webserverPort="80"
 webserverSecurePort="443">
```

```

sslKeyringLocation="path/to/sslkeyring"
sslStashfileLocation="path/to/stashfile"
sslCertlabel="definedbyuser"/>

```

```

<application type="war" id="myapp" name="myapp" location="{server.config.dir}/apps/myapp.war" />
<application type="war" id="snoop" name="snoop" location="{server.config.dir}/apps/snoop.war" />
</server>

```

**Note:**

- If you configure the web server plug-in to use SSL, you must enable the ssl-1.0 Liberty feature of Liberty.
- If the web server is using the default ports, you do not have to include the pluginConfiguration element in the server.xml file.
- The keystore that is used by the web server plug-in must be a CMS keystore, which can be created by using the Key Management (iKeyman) utility. You cannot use the JKS keystore that is created by Liberty or traditional for the web server plug-in, though you must exchange signer certificates between the web server plug-in keystore and the Liberty keystore.
- To configure the location of the plug-in log file, add the following code snippet to the server.xml file within the pluginConfiguration element:

```
logDirLocation="/path/to/log/file/"
```

2. Generate the plugin-cfg.xml file for your Liberty server and applications by calling the WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig MBean.

- a. Using the same Java SDK as the server, run the **jconsole** Java utility in a command window. For example, run the following command:

```
c:\java\bin\jconsole
```

The server process is listed in the choices that are waiting for connection.

**IBM i** For IBM i and z/OS, run the jconsole Java utility from a command window by using the Java SDK installed with the WCT on your workstation. For example, complete the following steps:

- 1) Create directory C:\restClient on your workstation.
- 2) Copy \${wlp.install.dir}/clients/restConnector.jar to the C:\restClient directory on your workstation.
- 3) Copy \${server.output.dir}/resources/security/key.jks to the C:\restClient directory on your workstation.
- 4) In a command window, type SET JAVA\_HOME=wct\_root\java.

**Note:** Ensure that you temporarily modify wct\_root\java\jre\lib\security\java.security by commenting out the two lines that set the SSL socket factories to the WebSphere Application Server SSL socket factories. This modification is documented in the Troubleshooting SSL section of the troubleshooting tips topic.

- 5) From the same command window, run the **jconsole** Java utility.

For example, run the following command:

```

"%JAVA_HOME%\bin\jconsole -J-Djava.class.path="%JAVA_HOME%\lib\jconsole.jar;"%JAVA_HOME%\lib\tools.jar;
C:\restClient\restConnector.jar -J-Djavax.net.ssl.trustStore=C:\restClient\key.jks
-J-Djavax.net.ssl.trustStorePassword=Liberty -J-Djavax.net.ssl.trustStoreType=jks

```

You might also need the following parameter:

```
-J-Dcom.ibm.ws.jmx.connector.client.disableURLHostnameVerification=true
```

- b. Connect to your server then click the **MBeans** tab. **IBM i** After the jConsole starts, select **Remote Process**, and enter the JMX service URL: service:jmx:rest://<host>:<port>/IBMJMXConnectorREST. The port number is the HTTPS port. You must also provide the user name and password.

- c. Locate the `com.ibm.ws.jmx.mbeans.generatePluginConfig` MBean under the **WebSphere** domain.
- d. Call the `generateDefaultPluginConfig` operation to generate the `plugin-cfg.xml` file, or call the `generatePluginConfig` operation to customize installation root directory and server name before you generate the `plugin-cfg.xml` file.

Here is an example of a `plugin-cfg.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config ASDisableNagle="false" AcceptAllContent="false" AppServerPortPreference="HostHeader"
 ChunkedResponse="false" FIPSEnable="false" IISDisableNagle="false"
 IISPluginPriority="High" IgnoreDNSFailures="false" RefreshInterval="60"
 ResponseChunkSize="64" SSLConsolidate="false" SSLPKCSDriver="REPLACE"
 SSLPKCSPassword="REPLACE" TrustedProxyEnable="false" VHostMatchingCompat="false">
 <Log LogLevel="Error" Name=".\\logs\\defaultServer\\http_plugin.log"/>
 <Property Name="ESIEnable" Value="true"/>
 <Property Name="ESIMaxCacheSize" Value="1024"/>
 <Property Name="ESIInvalidationMonitor" Value="false"/>
 <Property Name="ESIEnableToPassCookies" Value="false"/>
 <Property Name="PluginInstallRoot" Value="."/>
 <VirtualHostGroup Name="default_host">
 <VirtualHost Name="*:80"/>
 <VirtualHost Name="*:443"/>
 <VirtualHost Name="*:9080"/>
 </VirtualHostGroup>
 <ServerCluster CloneSeparatorChange="false" GetDWLMTable="false" IgnoreAffinityRequests="true"
 LoadBalance="Round Robin" Name="defaultServer_default_node_Cluster"
 PostBufferSize="64" PostSizeLimit="-1" RemoveSpecialHeaders="true"
 RetryInterval="60">
 <Server CloneID="b564bdc7-2c27-4a4b-ad37-9213c66e60d1" ConnectTimeout="0"
 ExtendedHandshake="false" MaxConnections="-1" Name="default_node_defaultServer0"
 ServerIOTimeout="900" WaitForContinue="false">
 <Transport Hostname="somehost.example.com" Port="9080" Protocol="http"/>
 </Server>
 <PrimaryServers>
 <Server Name="default_node_defaultServer0"/>
 </PrimaryServers>
 </ServerCluster>
 <UriGroup Name="default_host_defaultServer_default_node_Cluster_URIs">
 <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/myapp/*"/>
 <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/snoop/*"/>
 </UriGroup>
 <Route ServerCluster="defaultServer_default_node_Cluster"
 UriGroup="default_host_defaultServer_default_node_Cluster_URIs"
 VirtualHostGroup="default_host"/>
</Config>
```

The `plugin-cfg.xml` file is generated in the `${server.output.dir}` directory.

**Note:**

- You can use the **jConsole** utility with Liberty. However, any issues with the utility itself must be reported to your Java SDK provider.
  - The management interface for the `WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig` MBean is `com.ibm.websphere.webcontainer.GeneratePluginConfigMBean`.
3. Copy the `plugin-cfg.xml` file to the machine that hosts the web server. IBM i For IBM i, complete the following steps:
    - a. Run the **manageprofiles** Qshell command to create an http profile. For example, `plugins_root/bin/manageprofiles -create -profileName http -templatePath http`.
    - b. Copy the `plugin-cfg.xml` file to the config directory of the http profile that was created in substep a, for example, `plugin_profile_root/config/plugin-cfg.xml`, and set the file permissions.

- c. Run the following command from a system command line to change the file authorities to the required settings:

```
CHGAUT USER(QEJBSVR QTMHHTTP QNOTES) OBJ('plugin_profile_root/config/plugin-cfg.xml') DTAUT(*RWX)
```

## Configuring session persistence for Liberty

When session data must be maintained across a server restart or an unexpected server failure, you can configure Liberty to persist the session data to a database. This configuration allows multiple servers to share the same session data, and session data can be recovered in the event of a failover.

### About this task

To configure one or more servers in Liberty to persist session data to a database, complete the following steps.

### Procedure

1. Define a shared session management configuration that you can reuse among all of your servers. You must complete the following steps, as a minimum requirement:
  - a. Enable the `sessionDatabase-1.0` feature.
  - b. Define a data source:
 

```
<dataSource id="SessionDS" ... />
```
  - c. Refer to the data source from the session database configuration.
 

```
<httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS" ... />
```
  - d. Refer to the persistent storage location from the session management configuration.
 

```
<httpSession storageRef="SessionDB" ... />
```

**Note:** The `storageRef` attribute of the `httpSession` element and the `id` attribute of the `httpSessionDatabase` element are not mandatory. If the `sessionDatabase-1.0` feature is enabled and the `httpSessionDatabase` element references a valid data source, session persistence is enabled even if the `storageRef` attribute is not set.

See **\*\*\*\* MISSING FILE \*\*\*\*** for details about the `httpSession` and `httpSessionDatabase` elements.

For example, you can create a file named `${shared.config.dir}/httpSessionPersistence.xml` as follows:

```
<server description="Demonstrates HTTP Session Persistence Configuration">
 <featureManager>
 <feature>sessionDatabase-1.0</feature>
 <feature>servlet-3.0</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="${httpPort}">
 <tcpOptions soReuseAddr="true"/>
 </httpEndpoint>

 <fileset id="DerbyFiles" includes="*.jar" dir="${shared.resource.dir}/derby/client"/>
 <library id="DerbyLib" filesetRef="DerbyFiles"/>
 <jdbcDriver id="DerbyDriver" libraryRef="DerbyLib"/>
 <dataSource id="SessionDS" jdbcDriverRef="DerbyDriver">
 <properties.derby.client user="user1" password="password1"
 databaseName="${shared.resource.dir}/databases/SessionDB"
 createDatabase="create"/>
 </dataSource>

 <httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS"/>
 <httpSession storageRef="SessionDB" cloneId="${cloneId}"/>
</server>
```

```
<application id="test" name="test" type="ear" location="\${shared.app.dir}/test.ear"/>
</server>
```

**Note:** When multiple servers are configured to persist session data to the same database, those servers must share the same session management configuration. Any other configuration is not supported. For example, it is not possible for one server to use a multi-row schema while another server uses a single-row schema.

The HTTP server plugin uses the clone ID that is inserted into the response/request header to maintain session affinity between requests. While the clone ID is normally unchanging, in Liberty, the clone ID is generated when you start a server for the first time and it is regenerated if you start the server with the `--clean` option. For production use, manually assigning a clone ID will ensure that the ID is stable and that request affinity is correctly maintained. The clone ID must be unique for each server and can be 8 to 9 alphanumeric characters in length and is specified in step 3.

2. Include the shared session management configuration in each of your servers. For example, create two `server.xml` files for server instances named `s1` and `s2`, as follows:

```
• \${wlp.user.dir}/servers/s1/server.xml
• \${wlp.user.dir}/servers/s2/server.xml
<server description="Example Server">
 <include location="\${shared.config.dir}/httpSessionPersistence.xml"/>
</server>
```

See “Using include elements in configuration files” on page 969.

3. Specify unique variables in the `bootstrap.properties` file of each server.

```
• \${wlp.user.dir}/servers/s1/bootstrap.properties
 httpPort=9081
 cloneId=s1
• \${wlp.user.dir}/servers/s2/bootstrap.properties
 httpPort=9082
 cloneId=s2
```

4. Create a table for session persistence before you start the servers.

- If you want to change the default row size, table name, or table space name, see **\*\*\*\* MISSING FILE \*\*\*\*** for details about the `httpSessionDatabase` element.

- **Distributed operating systems** No additional action is required if your server is installed on one of the distributed operating systems. The server automatically creates the table.
- If your server is using DB2 for session persistence, you can increase the page size to optimize performance for writing large amounts of data to the database.

5. Synchronize the system clocks of all machines that host Liberty servers. If the system clocks are not synchronized, session invalidation can occur prematurely
6. Optional: If required, integrate HTTP sessions and security in Liberty. By default, after a session is created and accessed within a protected resource with security enabled, only the originating owner of that session can access it. Session security (security integration) is enabled by default.
7. Optional: If required, Install and configure the web server plug-in to route requests to each of the servers you configured. The session affinity is only maintained if your plug-in configuration specifies clone IDs that match the clone IDs defined in the server configuration.

## Configuring and deploying a basic JCA ResourceAdapter

### 0.5.5.2

You can configure and deploy a basic Java™ EE Connector Architecture (JCA) ConnectionFactory and Resource Adapter.

## About this task

You can install a resource adapter and configure instances of the resources it provides. This task uses an example resource adapter called ExampleRA.rar, which provides 3 types of resources: a connection factory and two types of administered objects.

### Procedure

1. Enable the JCA feature in your server.xml file. The server.xml file is found at [path\_to\_liberty\wlp\usr\servers\server\_name]

```
<server>
 <featureManager>
 <feature>jca-1.6</feature>
 <feature>servlet-3.0</feature>
 </featureManager>
</server>
```

2. Place the resource adapter RAR file (ExampleRA.rar) into the dropins folder of your server. If your server is running, you will see the following message in your console log indicating that the resource adapter has been installed:

```
[AUDIT] J2CA7001I: Resource adapter ExampleRA installed in 1.306 seconds.
```

3. Inspect the deployment descriptor, annotations, and other documentation from the resource adapter to identify which types of resources the adapter provides and the configuration properties that each adapter accepts. The example resource adapter, ExampleRA.rar, has this information in the ra.xml deployment descriptor. The ra.xml file is found at [path\_to\_ExampleRA\ExampleRA\META-INF.] The deployment descriptor identifies 3 types of resources you can configure.

```
<connection-definition>
 <managedconnectionfactory-class>com.ibm.example.jca.adapter.ManagedConnectionFactoryImpl</managedconnectionfactory-class>
 <config-property>
 <config-property-name>tableName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfactory-interface>
 ...
</connection-definition>
```

```
 <adminobject>
 <adminobject-interface>javax.resource.cci.ConnectionSpec</adminobject-interface>
 <adminobject-class>com.ibm.example.jca.adapter.ConnectionSpecImpl</adminobject-class>
 <config-property>
 <config-property-name>readOnly</config-property-name>
 <config-property-type>java.lang.Boolean</config-property-type>
 <config-property-value>>false</config-property-value>
 </config-property>
 </adminobject>
```

```
 <adminobject>
 <adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
 <adminobject-class>com.ibm.example.jca.adapter.InteractionSpecImpl</adminobject-class>
 <config-property>
 <description>Function name. Supported values are: ADD, FIND, REMOVE</description>
 <config-property-name>functionName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 </adminobject>
```

4. In the server.xml file, configure instances of the available resource types.

```
<server>
 <featureManager>
 <feature>jca-1.6</feature>
 <feature>servlet-3.0</feature>
 </featureManager>

 <connectionFactory jndiName="eis/conFactory">
```



```

 <properties.ExampleRA tableName="TABLE1"/>
</connectionFactory>

<adminObject jndiName="eis/conSpec">
 <properties.ExampleRA.ConnectionSpec/>
</adminObject>

<adminObject jndiName="eis/iSpec_ADD">
 <properties.ExampleRA.InteractionSpec functionName="ADD"/>
</adminObject>

<adminObject jndiName="eis/iSpec_FIND">
 <properties.ExampleRA.InteractionSpec functionName="FIND"/>
</adminObject>

</server>

```

5. Use resource injection to access the resources in your servlet; for example:

```

@Resource(lookup = "eis/conFactory")
private ConnectionFactory conFactory;

@Resource(lookup = "eis/conSpec")
private ConnectionSpec conSpec;

@Resource(lookup = "eis/iSpec_ADD")
private InteractionSpec iSpec_ADD;

@Resource(lookup = "eis/iSpec_FIND")
private InteractionSpec iSpec_FIND;

...

MappedRecord input = conFactory.getRecordFactory().createMappedRecord("input");
input.put("city", "Rochester");
input.put("state", "Minnesota");
input.put("population", 106769);

Connection con = conFactory.getConnection(conSpec);
try {
 Interaction interaction = con.createInteraction();
 interaction.execute(iSpec_ADD, input);
 interaction.close();
} finally {
 con.close();
}

```

**Note:** You must enable the JNDI feature in the `server.xml` file if you want to look up the resources from the namespace rather than using injection.

## Overview of JCA configuration elements

### 8.5.5.2

The Java Platform, Enterprise Edition Connector Architecture (JCA) feature provides configuration elements to define instances of connection factories, administered objects, and activation specifications, and to associate these instances with an installed resource adapter. Each of the JCA configuration elements consists of two basic parts, a top-level element and a subelement, both of which are required for the configured instance.

A *top-level element* configures general capabilities that are provided by the Liberty server, such as JNDI name, connection management, and container authentication. A *subelement* ties the instance to an installed resource adapter and enables you to specify vendor-defined configuration properties.

Generic JCA pre-defined top-level configuration elements:

- connectionFactory
- adminObject
- activationSpec

If the JMS feature is enabled, there are also pre-defined generic configuration elements for JMS:

- jmsConnectionFactory
- jmsQueueConnectionFactory
- jmsTopicConnectionFactory
- jmsDestination
- jmsQueue
- jmsTopic
- jmsActivationSpec

Subelements are generated from the resource adapter deployment descriptor and annotations when your resource adapter is installed. You will not see any documentation of the available subelements in the static documentation for available server configuration elements.

Use the following rules to generate the names of the server configuration subelements:

- If a resource adapter provides exactly one interface within one of the listed categories, such as connectionFactory or adminObject, the subelement is: `properties.<rar_identifier>`
- If the interface name is unique without the package name, the subelement is: `properties.<rar_identifier>.<InterfaceName>`
- If the implementation name is unique without the package name, the subelement is: `properties.<rar_identifier>.<ImplementationName>`
- In other cases, the subelement name is `properties.<rar_identifier>.<fully.qualified.InterfaceName>` or `properties.<rar_identifier>.<fully.qualified.ImplementationName>`

The following examples illustrate the case where only one interface within each category is provided by a resource adapter with identifier MyAdapter:

```
<connectionFactory jndiName="eis/cf1" containerAuthDataRef="auth1">
 <properties.MyAdapter portNumber="1234" someVendorProperty="100"/>
</connectionFactory>

<connectionFactory jndiName="eis/cf2" containerAuthDataRef="auth2">
 <properties.MyAdapter portNumber="1234" someVendorProperty="200"/>
</connectionFactory>

<jmsConnectionFactory jndiName="jms/cf">
 <properties.MyAdapter serverName="localhost" anotherProperty="40"/>
</jmsConnectionFactory>

<jmsQueueConnectionFactory jndiName="jms/qcf">
 <properties.MyAdapter serverName="localhost" vendorProp1="1"/>
</jmsQueueConnectionFactory>

<jmsTopicConnectionFactory jndiName="jms/tcf">
 <properties.MyAdapter serverName="localhost" prop1="A" prop2="B"/>
</jmsTopicConnectionFactory>

<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter functionName="find" executionTimeout="5000"/>
</adminObject>

<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter name="DEST1"/>
</jmsDestination>

<jmsQueue jndiName="jms/queue1">
 <properties.MyAdapter queueName="QUEUE1"/>
</jmsQueue>
```

```

<jmsTopic id="topic1" jndiName="jms/topic1">
 <properties.MyAdapter topicName="TOPIC1"/>
</jmsTopic>
<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter prop1="a" prop2="b" prop3="c"/>
</activationSpec>
<jmsActivationSpec id="app1/module1/MyJMSMessageDrivenBean">
 <properties.MyAdapter destinationRef="topic1"/>
</jmsActivationSpec>

```

## Configuring resource adapters: 8.5.5.2

You can configure resource adapters that comply with the Java EE Connector Architecture (JCA) specification versions 1.6, 1.5, or 1.0.

### About this task

You can install and configure a resource adapter and the various connection factories, administered objects, and activation specifications as defined in the JCA specification.

### Procedure

1. Update the `server.xml` file to add the `jca-1.6` feature under the `featureManager` tag. The `server.xml` file is found at `[path_to_liberty\wlp\usr\servers\server_name]`

```

<featureManager>
 <feature>jca-1.6</feature>
</featureManager>

```

2. (Optional) Enable the following additional features based on the needs of your system:
  - If your resource adapter provides JMS specification interfaces, enable the `jms-1.1` feature.
 

```
<feature>jms-1.1</feature>
```
  - If you want to look up connection factories and administered objects from your application, enable the `jndi-1.0` feature.
 

```
<feature>jndi-1.0</feature>
```
  - If your resource adapter provides activation specifications for message driven beans, enable the `mdb-3.1` feature.
 

```
<feature>mdb-3.1</feature>
```
  - If your resource adapter supports inbound security, enable the `jcaInboundSecurity-1.0` feature.
 

```
<feature>jcaInboundSecurity-1.0</feature>
```
  - If your resource adapter supports bean validation and you want your beans to be validated, you can enable the `beanValidation-1.0` feature.
 

```
<feature>beanValidation-1.0</feature>
```

3. Configure one or more resource adapters in the server. You can use one of the following methods to configure the resource adapter.

- Configure a standalone resource adapter by editing the `server.xml` file.

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

- Configure an embedded resource adapter by editing the `server.xml` file to install an application that embeds one or more resource adapter modules. The following example assumes that the `app1.ear` file contains one or more embedded RAR files:

```
<application location="C:/applications/app1.ear"/>
```

- Allow the server to automatically configure a standalone resource adapter by dropping the RAR file in the server drop-ins folder.

```
wlp/usr/servers/your-server-name/dropins/MyDropinAdapter.rar
```

- Allow the server to automatically configure an application containing one or more embedded resource adapters by dropping the EAR file in the server drop-ins folder. The following example assumes that the app2.ear file contains one or more embedded RAR files:

```
wlp/usr/servers/your-server-name/dropins/app2.ear
```

4. Start the application server. After the server is started, messages such as the following are displayed in the console.log file:

```
[AUDIT] J2CA7001I: Resource adapter MyAdapter installed in 0.495 seconds.
[AUDIT] J2CA7001I: Resource adapter MyDropinAdapter installed in 0.311 seconds.
[AUDIT] J2CA7001I: Resource adapter appl.MyEmbeddedAdapter installed in 0.247 seconds.
[AUDIT] J2CA7001I: Resource adapter app2.anotherEmbeddedAdapter installed in 0.518 seconds.
```

## Example

A unique identifier for a resource adapter is necessary to identify configured instances of connection factories, administered objects, and activation specifications as being associated with an installed resource adapter. For stand-alone resource adapters, the module name is used as the identifier. For resource adapters embedded in applications, the combination of the application name plus the module name (delimited by the period character) are used as the identifier.

- To specify properties for a stand-alone resource adapter using a `properties.MyAdapter` subelement that includes the resource adapter identifier, `MyAdapter`:

```
<resourceAdapter location="C:/adapters/MyAdapter.rar">
 <properties.MyAdapter logFile="{server.output.dir}/logs/myAdapter.log"/>
</resourceAdapter>
```

- To associate a connection factory with a stand-alone resource adapter using a `properties.MyAdapter` subelement that includes the resource adapter identifier, `MyAdapter`:

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
<connectionFactory jndiName="eis/cf">
 <properties.MyAdapter serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- To associate a connection factory with a resource adapter `MyEmbeddedAdapter`, which is enabled in the `appl` application, using a `properties.appl.MyEmbeddedAdapter` subelement:

```
<application location="C:/applications/app1.ear"/>
<connectionFactory jndiName="eis/cf">
 <properties.appl.MyEmbeddedAdapter serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- In some cases, the module name is not sufficiently unique to serve as the identifier. This might happen, for example, if you install two different versions of the same resource adapter. Alternately, the module name might be unique, but undesirable for use in configuration because it is lengthy or contains non-alphanumeric characters. You can override the resource adapter identifier by specifying the `id` attribute.

The following example demonstrates how to override the identifier for stand-alone resource adapters:

```
<resourceAdapter id="MyAdapterV1" location="C:/adapters/version-1.0/MyAdapter.rar"/>
<resourceAdapter id="MyAdapterV2" location="C:/adapters/version-2.0/MyAdapter.rar"/>
<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapterV1 serverName="localhost" portNumber="1234"/>
</connectionFactory>
<connectionFactory jndiName="eis/cf2">
 <properties.MyAdapterV2 serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- The following example demonstrates how to override the identifier for a resource adapter that is embedded in an application. The example changes the identifier to `MyEmbeddedRA`:

```
<application location="C:/applications/app1.ear">
 <resourceAdapter id="MyEmbeddedAdapter" alias="MyEmbeddedRA"/>
</application>
<connectionFactory jndiName="eis/cf">
 <properties.appl.MyEmbeddedRA serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- To compute the module name for embedded resource adapters, the `<module-name>` entry in the resource adapter deployment descriptor (`ra.xml`) takes precedence as the module name. For example, given the following definition in `ra.xml`:

```
<connector ...>
 <module-name>MyRARModule</module-name>
</connector>
```

the module name would be set to "MyRARModule".

If the module name is absent from the connector deployment descriptor, the short form of the URI referring to the resource adapter module in the application deployment descriptor (`application.xml`) is used. For example, given the following module definition in `application.xml`:

```
<module>
 <connector>connectors/MyRARModule.rar</connector>
</module>
```

the module name would be computed as "MyRARModule".

If multiple resource adapters are embedded in an application and define the same `<module-name>` value, the first one listed in `application.xml` uses that module name. The module names of the other connectors with that same conflicting name are calculated from the full form of the URI with all / (forward slash) characters converted to a period (.). For example, if two connectors were embedded in an application both containing the following definition in `ra.xml`:

```
<connector ...>
 <module-name>MyRARModule</module-name>
</connector>
```

and the following definitions in `application.xml`:

```
<module>
 <connector>subfolder1/connector1.rar</connector>
</module>
<module>
 <connector>subfolder2/connector2.rar</connector>
</module>
```

The module name for the first connector would be "MyRARModule" and the module name for the second would be "subfolder2.connector2.rar"

## Configuring JCA connection factories: 8.5.5.2

You can configure connection factories that comply with Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more connection factory instances for connection factory types that are provided by an installed resource adapter.

**Note:** To configure JCA support for the Liberty profile, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. This topic assumes that a resource adapter with a unique identifier of MyAdapter has already been configured in the server, see the documentation on configuring resource adapters for further details. An end-to-end example of configuring a basic scenario is provided in the following steps.

**Note:** Editing the properties sub-elements of the server configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere® Development Tools (WDT) is not supported.

### Procedure

1. Update the `server.xml` file to add the `jca-1.6` feature under the `featureManager` tag.

```

<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>

```

2. Install a resource adapter. For example, update the server.xml file as follows:

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. Configure one or more connection factory instances. When you configure the connection factory instances, you must supply a properties subelement, even if you do not want to override any configuration properties, in order to associate the connectionFactory element with a connection factory interface that is provided by a particular resource adapter. In the following example, the MyAdapter resource adapter provides only one type of connection factory:

```

<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapter portNumber="1234" someVendorProperty="100"/>
</connectionFactory>

```

```

<connectionFactory jndiName="eis/cf2" containerAuthDataRef="auth2">
 <connectionManager maxPoolSize="20" connectionTimeout="0"/>
 <properties.MyAdapter portNumber="1234" someVendorProperty="200"/>
</connectionFactory>
<authData id="auth2" user="user2" password="{xor}Lz4sLCgwLTtt"/>

```

4. (Optional) If required, identify the available connection factory property subelement names.

- If a resource adapter provides exactly one connection factory interface, excluding any JMS connection factories, the subelement is: `properties.<rar_identifier>`
- If the interface name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<InterfaceName>`
- If the implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ImplementationName>`
- In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.InterfaceName>`

## Example

Use the following example to learn how to configure resource adapters with two connection factories with unique interface class names.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two connection factories with unique interface class names:

```

<connection-definition>
 <config-property>
 <config-property-name>ServerName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</connectionfactory-impl-class>
</connection-definition>
<connection-definition>
 <config-property>
 <config-property-name>ServerName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.sql.DataSource</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.DataSourceImpl</connectionfactory-impl-class>
</connection-definition>

```

The following is an example of a server configuration for this scenario:

```

<connectionFactory jndiName="eis/cf">
 <properties.MyAdapter.ConnectionFactory serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="jdbc/ds">
 <properties.MyAdapter.DataSource serverName="localhost"/>
</connectionFactory>

```

Use the following example to learn how to configure resource adapters with two connection factories with unique implementation class names.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two connection factories with unique implementation class names:

```

<connection-definition>
 <config-property>
 <config-property-name>ServerName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</connectionfactory-impl-class>
</connection-definition>

<connection-definition>
 <config-property>
 <config-property-name>ServerName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>com.vendor.adapter.ConnectionFactory</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.MyConnectionFactoryImpl</connectionfactory-impl-class>
</connection-definition>

```

The following is an example of a server configuration for this scenario:

```

<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapter.ConnectionFactoryImpl serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/cf2">
 <properties.MyAdapter.MyConnectionFactoryImpl serverName="localhost"/>
</connectionFactory>

```

Use the following example to learn how to configure resource adapters with two connection factories where neither the simple interface nor implementation class names are unique.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two connection factories where neither the simple interface nor the implementation class names are unique:

```

<connection-definition>
 <config-property>
 <config-property-name>ServerName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</connectionfactory-impl-class>
</connection-definition>

<connection-definition>
 <config-property>
 <config-property-name>HostName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>com.vendor.adapter.custom.ConnectionFactory</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.custom.ConnectionFactoryImpl</connectionfactory-impl-class>
</connection-definition>

```

The following is an example of a server configuration for this scenario:

```
<connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.javax.resource.cci.ConnectionFactory serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.com.vendor.adapter.custom.ConnectionFactory hostName="localhost"/>
</connectionFactory>
```

It is possible to override the suffixes of configuration element names. See the information about customizing JCA configuration elements to learn how to override the suffixes of configuration element names.

## Configuring JCA administered objects: 8.5.5.2

You can configure administered objects that comply with the Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more instances of administered objects that are provided by an installed resource adapter.

**Note:** To configure JCA support for Liberty, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. This topic assumes that a resource adapter with a unique identifier of `MyAdapter` has already been configured in the server, see the documentation on configuring resource adapters for further details. An end-to-end example of configuring a basic scenario is provided in the following steps.

**Note:** Editing the properties sub-elements of the server configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere® Development Tools (WDT) is not supported.

### Procedure

1. Update the `server.xml` file to add the `jca-1.6` feature under the `featureManager` tag.

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>
```

2. Install a resource adapter. For example, update the `server.xml` file as follows:

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. Configure one or more administered object instances. When you configure administered object instances, you must supply a `properties` subelement, even if you do not want to override any configuration properties, to associate the `adminObject` element with an administered object type that is provided by a particular resource adapter. In the following example, the `MyAdapter` resource adapter provides only one type of administered object:

```
<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter functionName="find" executionTimeout="5000"/>
</adminObject>
```

4. (Optional) If required, identify the available administered object property subelement names.
  - If a resource adapter provides exactly one administered object interface excluding any JMS destinations, queues and topics, the subelement name is: `properties.<rar_identifier>`



- If the interface name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<InterfaceName>`
- If the implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ImplementationName>`
- If the combination of interface name and implementation name are unique without the package name, the subelement name is: `properties.<rar_identifier>.<InterfaceName>-<ImplementationName>`
- In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.InterfaceName>-<fully.qualified.ImplementationName>`

## Example

Use the following example to learn how to configure resource adapters with two administered objects with unique interface class names.

In the following snippet from a `ra.xml` file, the `MyAdapter` resource adapter provides two administered objects with unique interface class names:

```
<adminobject>
<adminobject-interface>javax.resource.cci.ConnectionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.ConnectionSpecImpl</adminobject-class>
<config-property>
 <config-property-name>isolationLevel</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.InteractionSpecImpl</adminobject-class>
<config-property>
 <config-property-name>FunctionName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>
```

The following is an example of a server configuration for this scenario:

```
<adminObject jndiName="eis/connectionSpec">
 <properties.MyAdapter.ConnectionSpec isolationLevel="4"/>
</adminObject>

<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter.InteractionSpec functionName="find"/>
</adminObject>
```

Use the following example to learn how to configure resource adapters with two administered objects with unique implementation class names.

In the following snippet from a `ra.xml` file, the `MyAdapter` resource adapter provides two administered objects with unique implementation class names:

```
<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.FinderInteractionSpec</adminobject-class>
<config-property>
 <config-property-name>ResultSetType</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
```

```

</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.UpdaterInteractionSpec</adminobject-class>
<config-property>
 <config-property-name>ExecutionTimeout</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
</config-property>
...
</adminobject>

```

The following is an example of a server configuration for this scenario:

```

<adminObject jndiName="eis/finder">
 <properties.MyAdapter.FinderInteractionSpec resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/updater">
 <properties.MyAdapter.UpdaterInteractionSpec executionTimeout="3000"/>
</adminObject>

```

Use the following example to learn how to configure resource adapters with two administered objects where neither the simple interface nor implementation class names are unique.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two administered objects where neither the simple interface nor the implementation class names are unique:

```

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.finder.InteractionSpecImpl</adminobject-class>
<config-property>
 <config-property-name>ResultSetType</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.updater.InteractionSpecImpl</adminobject-class>
<config-property>
 <config-property-name>ExecutionTimeout</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
</config-property>
...
</adminobject>

```

The following is an example of a server configuration for this scenario:

```

<adminObject jndiName="eis/finder">
 <properties.MyAdapter.javax.resource.cci.InteractionSpec-com.vendor.adapter.finder.InteractionSpecImpl resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/updater">
 <properties.MyAdapter.javax.resource.cci.InteractionSpec-com.vendor.adapter.updater.InteractionSpecImpl executionTimeout="3000"/>
</adminObject>

```

It is possible to override the suffixes of configuration element names. See the information about customizing JCA configuration elements to learn how to override the suffixes of configuration element names.

## Configuring JCA activation specifications: 8.5.5.2

You can configure activation specifications that comply with the Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more instances of activation specifications that are provided by an installed resource adapter.

**Note:** To configure JCA support for Liberty, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. This topic assumes that a resource adapter with a unique identifier of `MyAdapter` has already been configured in the server, see the documentation on configuring resource adapters for further details. An end-to-end example of configuring a basic scenario is provided in the following steps.

**Note:** Editing the properties sub-elements of the server configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere® Development Tools (WDT) is not supported.

### Procedure

1. Update the `server.xml` file to add the `jca-1.6` feature under the `featureManager` tag.

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered ob
 ...
</featureManager>
```

2. Install a resource adapter. For example, update the `server.xml` file as follows:

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. Configure one or more activation specifications. When you configure activation specifications, you must supply a `properties` subelement, even if you do not want to override any configuration properties, to associate the `activationSpec` element with a message listener type that is provided by a particular resource adapter. In the following example, the `MyAdapter` resource adapter provides only one type of message listener:

```
<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter messageFilter="ALL"/>
</activationSpec>
```

4. If required, identify the available activation specification property subelement names.
  - If a resource adapter provides exactly one message listener interface, excluding any JMS connection factories, the subelement name is: `properties.<rar_identifier>`
  - If the message listener interface name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<MessageListenerInterfaceName>`
  - If the activation specification implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ActivationSpecificationImplementationName>`
  - If the activation specification implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ActivationSpecificationImplementationName>`
  - In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.MessageListenerInterfaceName>`
5. See the documentation on deploying message-driven beans for information about how to associate the activation specification with a message-driven bean.

## Example

Use the following example to learn how to configure resource adapters with two message listener types with unique interface class names.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two message listener types with unique interface class names:

```
<messageListener>
<messageListener-type>javax.resource.cci.MessageListener</messageListener-type>
<activationSpec>
 <activationSpec-class>com.vendor.adapter.CCIActivationSpec</activationSpec-class>
 <config-property>
 <config-property-name>maxSize</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
 </config-property>
 ...
</activationSpec>
...
</messageListener>

<messageListener>
<messageListener-type>com.vendor.adapter.MyMessageListener</messageListener-type>
<activationSpec>
 <activationSpec-class>com.vendor.adapter.MyActivationSpec</activationSpec-class>
 <config-property>
 <config-property-name>messageFilter</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 ...
</activationSpec>
...
</messageListener>
```

The following is an example of a server configuration for this scenario:

```
<activationSpec id="app1/module1/CCIMessageDrivenBean">
 <properties.MyAdapter.MessageListener maxSize="1024"/>
</activationSpec>

<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter.MyMessageListener messageFilter="ALL"/>
</activationSpec>
```

Use the following example to learn how to configure resource adapters with two message listener types with unique implementation class names

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two message listener types with unique implementation class names:

```
<messageListener>
<messageListener-type>javax.resource.cci.MessageListener</messageListener-type>
<activationSpec>
 <activationSpec-class>com.vendor.adapter.CCIActivationSpec</activationSpec-class>
 <config-property>
 <config-property-name>maxSize</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
 </config-property>
 ...
</activationSpec>
...
</messageListener>

<messageListener>
<messageListener-type>com.vendor.adapter.MessageListener</messageListener-type>
<activationSpec>
```

```

<activationSpec id="app1/module1/CCIActivationSpec">
 <properties>
 <property name="messageFilter" value="ALL" />
 </properties>
</activationSpec>

```

The following is an example of a server configuration for this scenario:

```

<activationSpec id="app1/module1/CCIActivationSpec">
 <properties>
 <property name="CCIActivationSpec" value="maxSize=1024" />
 </properties>
</activationSpec>

<activationSpec id="app1/module1/MyActivationSpec">
 <properties>
 <property name="MyActivationSpec" value="messageFilter=ALL" />
 </properties>
</activationSpec>

```

Use the following example to learn how to configure resource adapters with two message listener types where neither the simple interface nor implementation class names are unique.

In the following snippet from a ra.xml file, the MyAdapter resource adapter provides two message listener types where neither the simple interface nor the implementation class names are unique:

```

<messageListener>
 <messageListenerType>javax.resource.cci.MessageListener</messageListenerType>
 <activationSpec>
 <activationSpecClass>com.vendor.adapter.cci.ActivationSpec</activationSpecClass>
 <configProperty name="maxSize" type="java.lang.Long" value="1024" />
 </activationSpec>
</messageListener>

<messageListener>
 <messageListenerType>com.vendor.adapter.MessageListener</messageListenerType>
 <activationSpec>
 <activationSpecClass>com.vendor.adapter.ActivationSpec</activationSpecClass>
 <configProperty name="messageFilter" type="java.lang.String" value="ALL" />
 </activationSpec>
</messageListener>

```

The following is an example of a server configuration for this scenario:

```

<activationSpec id="app1/module1/CCIActivationSpec">
 <properties>
 <property name="CCIActivationSpec" value="javax.resource.cci.MessageListener maxSize=1024" />
 </properties>
</activationSpec>

<activationSpec id="app1/module1/MyActivationSpec">
 <properties>
 <property name="MyActivationSpec" value="com.vendor.adapter.MessageListener messageFilter=ALL" />
 </properties>
</activationSpec>

```

It is possible to override the suffixes of configuration element names. See the information about customizing JCA configuration elements to learn how to override the suffixes of configuration element names.

## Configuring JMS connection factories: 8.5.5.2

You can configure JMS connection factories that are provided by resource adapters that comply with the Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more JMS connection factory instances for JMS connection factory types that are provided by an installed resource adapter.

Configuration elements are provided for the following types of JMS connection factories:

- `javax.jms.ConnectionFactory`: `jmsConnectionFactory`
- `javax.jms.QueueConnectionFactory`: `jmsQueueConnectionFactory`
- `javax.jms.TopicConnectionFactory`: `jmsTopicConnectionFactory`

### Note:

To add JCA support for Liberty, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. Editing portions of the configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view is not supported in the Beta.

### Procedure

Configure one or more JMS connection factory instances. When you configure the connection factory instances, you must supply a `properties` subelement, even if you do not want to override any configuration properties, to associate the `jmsConnectionFactory`, `jmsQueueConnectionFactory`, or `jmsTopicConnectionFactory` element with a connection factory interface that is provided by a particular resource adapter. The `properties` subelement always follows the pattern `properties.<rar_identifier>` for JMS connection factories. In the following example, the `MyAdapter` resource adapter provides only one type of connection factory:

```
<jmsConnectionFactory jndiName="jms/cf" containerAuthDataRef="auth1">
 <properties.MyAdapter serverName="localhost" anotherProperty="40"/>
</jmsConnectionFactory>
<authData id="auth1" user="user1" password="{xor}Lz4sLCgwLTtu"/>

<jmsQueueConnectionFactory jndiName="jms/qcf">
 <connectionManager maxPoolSize="20" connectionTimeout="0"/>
 <properties.MyAdapter serverName="localhost" vendorProp1="1"/>
</jmsQueueConnectionFactory>

<jmsTopicConnectionFactory jndiName="jms/tcf">
 <properties.MyAdapter serverName="localhost" prop1="A" prop2="B"/>
</jmsTopicConnectionFactory>
```

**Note:** This topic assumes that a resource adapter with a unique identifier of `MyAdapter` has already been configured in the server and that the `jms-1.1` feature has been enabled. See topic “Configuring resource adapters” on page 995 for further details.

**Limitation:** Editing the properties of the resource adapter configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere Development Tools (WDT) is not supported.

## Configuring JMS destinations: 8.5.5.2

You can configure JMS destinations that are provided by resource adapters that comply with the Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more instances of JMS destination, queue, or topic types that are provided by an installed resource adapter.

Configuration elements are provided for the following types of JMS destinations:

- `javax.jms.Destination`: `jmsDestination`
- `javax.jms.Queue`: `jmsQueue`
- `javax.jms.Topic`: `jmsTopic`

### Note:

To add JCA support for Liberty, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. Editing portions of the configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view is not supported in the Beta.

### Procedure

1. Configure one or more JMS destination, queue, or topic instances. When you configure the destination instances, you must supply a `properties` subelement, even if you do not want to override any configuration properties, to associate the `jmsDestination`, `jmsQueue`, or `jmsTopic` element with a JMS destination interface that is provided by a particular resource adapter. In the following example, the `MyAdapter` resource adapter provides only one type of JMS destination, one type of JMS queue, and one type of JMS topic:

```
<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter name="DEST1"/>
</jmsDestination>
```

```
<jmsQueue jndiName="jms/queue1">
 <properties.MyAdapter queueName="QUEUE1"/>
</jmsQueue>
```

```
<jmsTopic id="topic1" jndiName="jms/topic1">
 <properties.MyAdapter topicName="TOPIC1"/>
</jmsTopic>
```

2. (Optional) If required, identify the available destination, queue, and topic property subelement names.

**Note:** This topic assumes that a resource adapter with a unique identifier of `MyAdapter` has already been configured in the server. See topic “Configuring resource adapters” on page 995 for further details.

Limitation: Editing the properties of the resource adapter configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere Development Tools (WDT) is not supported.

- If a resource adapter provides exactly one type of administered object with the `javax.jms.Destination` interface, the subelement name is: `properties.<rar_identifier>`
- If the implementation name is unique without the package name the subelement name is: `properties.<rar_identifier>.<ImplementationName>`
- In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.InterfaceName>`

- If a resource adapter provides exactly one type of administered object with the `javax.jms.Queue` interface, the subelement name is: `properties.<rar_identifier>`
- If the implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ImplementationName>`
- In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.InterfaceName>`
- If a resource adapter provides exactly one type of administered object with the `javax.jms.Topic` interface, the subelement name is: `properties.<rar_identifier>`
- If the implementation name is unique without the package name, the subelement name is: `properties.<rar_identifier>.<ImplementationName>`
- In other cases, the subelement name is: `properties.<rar_identifier>.<fully.qualified.InterfaceName>`

## Example

Use the following example to learn how to configure resource adapters with two JMS destinations with unique implementation class names

In the following snippet from a `ra.xml` file the `MyAdapter` resource adapter provides two JMS destinations with unique implementation class names:

```
<adminobject>
<adminobject-interface>javax.jms.Destination</adminobject-interface>
<adminobject-class>com.vendor.adapter.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>queueName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.jms.Destination</adminobject-interface>
<adminobject-class>com.vendor.adapter.TopicImpl</adminobject-class>
<config-property>
 <config-property-name>topicName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>
```

The following is an example of a server configuration for this scenario:

```
<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter.QueueImpl queueName="D1"/>
</adminObject>

<jmsDestination jndiName="jms/destination2">
 <properties.MyAdapter.TopicImpl topicName="D2"/>
</jmsDestination>
```

Use the following example to learn how to configure resource adapters with two administered objects without implementation class names that are unique.

In the following snippet from a `ra.xml` file the `MyAdapter` resource adapter provides two administered objects with non-unique implementation class names:

```
<adminobject>
<adminobject-interface>javax.jms.Queue</adminobject-interface>
<adminobject-class>com.vendor.adapter.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>queueName</config-property-name>
```



```

 <config-property-type>java.lang.String</config-property-type>
 </config-property>
 ...
</adminobject>

<adminobject>
<adminobject-interface>javax.jms.Queue</adminobject-interface>
<adminobject-class>com.vendor.adapter.advanced.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>name</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
 ...
</adminobject>

```

The following is an example of a server configuration for this scenario:

```

<jmsQueue jndiName="jms/myQueue">
 <properties.MyAdapter.com.vendor.adapter.QueueImpl queueName="Q1"/>
</jmsQueue>

<jmsQueue jndiName="jms/myAdvancedQueue">
 <properties.MyAdapter.com.vendor.adapter.advanced.QueueImpl name="Q1"/>
</jmsQueue>

```

In some scenarios, lengthy configuration element names might be undesirable. See the information about customizing JCA configuration elements to learn how to override the suffixes of configuration element names.

## Configuring JMS activation specifications: 8.5.5.2

You can configure JMS activation specifications that are provided by resource adapters that comply with the Java EE Connector Architecture (JCA) specification.

### About this task

You can configure one or more JMS activation specification instances for JMS message listeners that are provided by an installed resource adapter.

#### Note:

To add JCA support for Liberty, you must edit the `server.xml` file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. Editing portions of the configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view is not supported in the Beta.

### Procedure

Configure one or more JMS activation specification instances. When you configure the activation specification instances, you must supply a `properties` subelement, even if you do not want to override any configuration properties, to associate the `jmsActivationSpec` element with a JMS message listener that is provided by a particular resource adapter. The `properties` subelement always follows the pattern `properties, <rar_identifier>` for JMS activation specifications. The following example configures two instances of JMS activation specifications:

```

<jmsActivationSpec id="app1/module1/MyJMSMessageDrivenBean">
 <properties.MyAdapter destinationRef="topic1"/>
</jmsActivationSpec>

```

```

<jmsActivationSpec id="app1/module1/AnotherJMSMessageDrivenBean">
 <containerAuthData user="user1" password="{xor}Lz4sLCgwLTtu"/>
 <properties.MyAdapter destinationRef="queue1"/>
</jmsActivationSpec>

```

**Note:** This topic assumes that a resource adapter with a unique identifier of MyAdapter has already been configured in the server and that the jms-1.1 and mdb-3.1 features have been enabled. See topic “Configuring resource adapters” on page 995 for further details.

Limitation: Editing the properties of the resource adapter configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view of WebSphere Development Tools (WDT) is not supported.

## Customizing JCA configuration elements: 8.5.5.2

You can customize how JCA properties subelements are generated when installing a resource adapter.

### About this task

When you install a stand-alone resource adapter or a resource adapter that is embedded in an application, you can add one or more <customize> subelements under the <resourceAdapter> element to choose the suffix that is used for the properties subelement for the specified interface or implementation class. Customizing subelement enables you to avoid lengthy properties subelement names that might otherwise be required for the configuration elements to have unique names.

### Note:

To add JCA support for Liberty, you must edit the server.xml file using either the Source view of the Server configuration editor of the WebSphere® Application Server Developer Tools for Eclipse, or some other text editor. Editing portions of the configuration for connection factories, administrative objects, activation specifications, and resource adapters in the Design view is not supported in the Beta.

### Procedure

1. For a stand-alone resource adapter, start with the existing configuration that you want to customize. For example, if a resource adapter MyAdapter provides two connection factories, where neither the simple interface nor implementation class names are unique:

```

<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>
<resourceAdapter location="C:/adapters/MyAdapter.rar">

 <connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.javaax.resource.cci.ConnectionFactory serverName="localhost"/>
 </connectionFactory>

 <connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.com.vendor.adapter.custom.ConnectionFactory hostName="localhost"/>
 </connectionFactory>

```

2. Add customize subelements to the resourceAdapter to choose the suffixes for both of the connection factory interfaces.

```

<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>

<resourceAdapter location="C:/adapters/MyAdapter.rar">
 <customize interface="javaax.resource.cci.ConnectionFactory" suffix="cci"/>

```

```
<customize interface="com.vendor.adapter.custom.ConnectionFactory" suffix="custom"/>
</resourceAdapter>
```

```
<connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.cci serverName="localhost"/>
</connectionFactory>
```

```
<connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.custom hostName="localhost"/>
</connectionFactory>
```

3. For a resource adapter that is embedded in an application, start with the existing configuration that you want to customize. For example, assume that you have an application `app1` with an embedded resource adapter named `MyAdapter` as follows:

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>
```

```
<application name="app1" type="ear" location="C:/applications/app1.ear"/>
```

```
<adminObject jndiName="eis/interactionSpec-find">
 <properties.app1.MyAdapter.javaax.resource.cci.InteractionSpec-com.vendor.adapter.finder.InteractionSpecImpl resultSetType="ResultSetType.RESULT_SET_TYPE_SCROLL_INSENSITIVE_CURSOR_SENSITIVE"/>
</adminObject>
```

```
<adminObject jndiName="eis/interactionSpec-update">
 <properties.app1.MyAdapter.com.vendor.adapter.InteractionSpec-com.vendor.adapter.updater.InteractionSpecImpl executionTimeout="3000"/>
</adminObject>
```

4. Specify a `resourceAdapter` element for the Resource Adapter Archive (RAR) module in the application. Specify the `id` attribute to be the module name of the RAR module. Add `customize` subelements to choose the suffixes for both of the administered objects that are based on the interface or implementation class. In this example, only the implementation class is specified, which is sufficient to identify the administered objects:

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature> <!-- Add the jndi feature to enable look up of connection factories and administered objects -->
 ...
</featureManager>
```

```
<application name="app1" type="ear" location="C:/applications/app1.ear">
```

```
 <resourceAdapter id="MyAdapter">
 <customize implementation="com.vendor.adapter.finder.InteractionSpecImpl" suffix="finder"/>
 <customize implementation="com.vendor.adapter.updater.InteractionSpecImpl" suffix="updater"/>
 </resourceAdapter>
</application>
```

```
<adminObject jndiName="eis/interactionSpec-find">
 <properties.app1.MyAdapter.finder resultSetType="1003"/>
</adminObject>
```

```
<adminObject jndiName="eis/interactionSpec-update">
 <properties.app1.MyAdapter.updater executionTimeout="3000"/>
</adminObject>
```

## Accessing stand-alone resource adapters from Java EE applications: 8.5.5.2

You can access stand-alone resource adapters from Java EE applications.

### About this task

Stand-alone resource adapter classes and resources can be shared across multiple Java EE applications. By default, Java EE applications have access to JCA spec API but do not have access to the vendor classes

and resources of stand-alone resource adapters. A prerequisite of enabling this access is that both the resource adapter and the application must be configured in the server configuration.

In the following example, an application called Scholar and an application called Student are running on a server called Academy. Both applications need access to a resource adapter called Socrates16, which is provided in the socrates.rar file that is located in the C:/adapters/version-1.6 directory.

### Procedure

1. Configure the stand-alone resource adapter.

In the server.xml file, configure the stand-alone resource adapter by adding the following code:

```
<resourceAdapter id="Socrates16" location="C:/adapters/version-1.6/socrates.rar" />
```

2. Reference the resource adapter from the applications so that both applications can access the classes and resources that are provided in the resource adapter module.

In the server.xml file, set the classProviderRef attribute to the ID of the resource adapter within the class loading configurations of the applications by adding the following code:

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader classProviderRef="Socrates16" />
</application>
```

```
<application id="student" name="Student" type="ear" location="student.ear">
 <classloader classProviderRef="Socrates16" />
</application>
```

3. Optional: Configure the class loading of the stand-alone resource adapter to access third-party APIs.

By default, neither resource adapters nor Java applications can access third-party APIs. Whenever the class loading configuration of an application requires access to third-party APIs and the application requires access to a stand-alone resource adapter, configure the class loading of the resource adapter to also access third-party APIs.

In the server.xml file, configure the apiTypeVisibility attribute of the class loading configuration of the resource adapter to access third-party APIs by adding the following code:

```
<resourceAdapter id="Socrates16" location="C:/adapters/version-1.6/socrates.rar">
 <classloader apiTypeVisibility="spec, ibm-api, api, third-party" />
</resourceAdapter/>
```

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
 <classloader classProviderRef="Socrates16" apiTypeVisibility="spec, ibm-api, api, third-party" />
</application>
```

```
<application id="student" name="Student" type="ear" location="student.ear">
 <classloader classProviderRef="Socrates16" apiTypeVisibility="spec, ibm-api, api, third-party" />
</application>
```

## Configuring ManagedExecutorService instances

You can configure ManagedExecutorService instances to run asynchronous tasks with the specified thread context. It is a best practice for Java EE applications to avoid directly managing their own threads; therefore, the ManagedExecutorService extends the JSE ExecutorService to provide a way to launch asynchronous tasks within an application server environment. You might also configure the ManagedExecutorService to propagate various thread contexts that are relevant to Java EE applications to the thread of the asynchronous task.

### About this task

The ManagedExecutorService is available under the <concurrent-1.0> feature and enabled in the server.xml file as follows:

```
<featureManager>
 <feature>concurrent-1.0</feature>
</featureManager>
```

Propagation of context to the thread of a task that is executed by the `ManagedExecutorService` is managed by the context service. A default instance of the context service (`DefaultContextService`) is created by the server and configured to propagate at least `classloaderContext`, `jeeMetadataContext` and `securityContext`. This default context service instance is used if a `ManagedExecutorService` is created without referring to a specific context service instance or configuring a context service instance directly within. Refer to the [Configuring thread context service instances](#) topic for more information.

A default managed executor instance (`DefaultManagedExecutorService`) is available as `java:comp/DefaultManagedExecutorService` and uses the default context service instance for thread context capture and propagation.

## Procedure

Example configuration in the `server.xml` file:

- Managed executor service instance that is registered in JNDI with the name `concurrent/execSvc`, and that uses the default context service instance:

```
<managedExecutorService jndiName="concurrent/execSvc"/>
```

- Managed executor service instance with context service configured to propagate `jeeMetadataContext` only:

```
<managedExecutorService jndiName="concurrent/execSvc1">
 <contextService>
 <jeeMetadataContext/>
 </contextService>
</managedExecutorService>
```

- Managed executor service instance with `classloaderContext` and `securityContext`:

```
<managedExecutorService jndiName="concurrent/execSvc2">
 <contextService>
 <classloaderContext/>
 <securityContext/>
 </contextService>
</managedExecutorService>
```

- Thread context service that is shared by multiple managed executor service instances:

```
<contextService id="contextSvc1">
 <jeeMetadataContext/>
</contextService>
```

```
<managedExecutorService jndiName="concurrent/execSvc3" contextServiceRef="contextSvc1"/>
```

```
<managedExecutorService jndiName="concurrent/execSvc4" contextServiceRef="contextSvc1"/>
```

- Thread context service that inherits from the previous example and is used by a managed executor service instance:

```
<contextService id="contextSvc2" baseContextRef="contextSvc1">
 <classloaderContext/>
</contextService>
```

```
<managedExecutorService jndiName="concurrent/execSvc5" contextServiceRef="contextSvc2"/>
```

- Managed executor service instance with `zosWLMContext` plus thread context propagation inherited from the default context service instance:

```
<managedExecutorService jndiName="concurrent/execSvc6">
 <contextService baseContextRef="DefaultContextService">
 <zosWLMContext defaultTransactionClass="TRAN1"/>
 </contextService>
</managedExecutorService>
```

## Example

Managed executor serviced instances can be injected into application components (by using `@Resource`) or looked up with resource environment references (`resource-env-ref`). Regardless of how the instance is obtained, you can use it interchangeably as `javax.enterprise.concurrent.ManagedExecutorService` or its `java.util.concurrent.ExecutorService` superclass.

- Example that looks up the default managed executor:

```
ManagedExecutorService executor =
 (ManagedExecutorService) new InitialContext().lookup(
 "java:comp/DefaultManagedExecutorService");
executor.submit(doSomethingInParallel);
```

- Example that uses `@Resource` to inject as `java.util.concurrent.ExecutorService`:

```
@Resource(lookup="concurrent/execSvc1")
ExecutorService execSvc1;
```

...

```
// submit task to run
Future<Integer> future1 = execSvc1.submit(new Callable<Integer>() {
 public Integer call() throws Exception {
 // java:comp lookup is possible because <jeeMetadataContext> is configured
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ... make updates to the database
 return updateCount;
 }
});
Future<Integer> future2 = execSvc1.submit(anotherTaskThatUpdatesADatabase);
```

```
numUpdatesCompleted = future1.get() + future2.get();
```

- Example that uses `@Resource` to inject as `javax.enterprise.concurrent.ManagedExecutorService`:

```
@Resource(lookup="concurrent/execSvc1")
ManagedExecutorService execSvc1;
```

...

```
// submit task to run
Future<Integer> future1 = execSvc1.submit(new Callable<Integer>() {
 public Integer call() throws Exception {
 // java:comp lookup is possible because <jeeMetadataContext> is configured
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ... make updates to the database
 return updateCount;
 }
});
Future<Integer> future2 = execSvc1.submit(anotherTaskThatUpdatesADatabase);
```

```
numUpdatesCompleted = future1.get() + future2.get();
```

- Example `<resource-env-ref>` for `java.util.concurrent.ExecutorService` in the `web.xml` file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/execSvc2</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- Example `<resource-env-ref>` for `javax.enterprise.concurrent.ManagedExecutorService` in the `web.xml` file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/execSvc2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- Example lookup that uses a resource environment reference:

```
ExecutorService execSvc2 =
 (ExecutorService) new InitialContext().lookup("java:comp/env/concurrent/execSvc2");
```

```
futures = execSvc2.invokeAll(Arrays.asList(task1, task2, task3));
```

- Example lookup that uses a resource environment reference and casts to ManagedExecutorService:

```
ManagedExecutorService execSvc2 =
 (ManagedExecutorService) new InitialContext().lookup("java:comp/env/concurrent/execSvc2");
```

```
futures = execSvc2.invokeAll(Arrays.asList(task1, task2, task3));
```

## Configuring thread context service instances

8.5.5.4


You can configure ContextService instances to capture a managed thread context and apply it to invocations of specified interface methods on any thread.

### About this task

It is a best practice for Java EE applications to avoid directly managing their own threads; therefore, the ContextService provides a way to establish a previously captured thread context onto unmanaged threads, as well as managed threads, overlaying any thread context that is in place.

A default thread context service instance (DefaultContextService) is created by the server and configured to capture and propagate at least classloaderContext, jeeMetadataContext and securityContext. You can configure thread context propagation to include the following types of thread context:

#### classloaderContext

Makes the thread context classloader of the submitter of the task available to the task. 

**8.5.5.4** If the context classloader is serialized, the classloader must be a thread context classloader from the application. Classloader serialization for Web Application Bundles is not currently supported.

#### jeeMetadataContext

Makes the namespace of the application component that submitted the task available to the task.

#### securityContext

**8.5.5.4** You must enable the appSecurity-2.0 feature in the server.xml file to use this type of thread context. Makes the caller subject and invocation subject of the submitter of the task available to the task, and this is accomplished by logging in with the submitter's WSPincipal using JAAS login. For details on what information in the submitter's subject is not in the security context, see the concurrent-1.0 feature restrictions.

**Important:** Additional thread context providers might be made available by features in stack products. The optional baseContextRef attribute allows a context service instance to inherit from the context configuration of another context service instance.

### Procedure

Enable the thread context service in the server.xml file. The thread context service is available under the <concurrent-1.0> feature.

```
<featureManager>
 <feature>concurrent-1.0</feature>
</featureManager>
```

### Example

Configure thread context service instances in the server.xml file:

- Thread context service that is registered in JNDI with the name, `concurrent/threadContextSvc1`, that captures and propagates `jeeMetadataContext` only:

```
<contextService id="threadContextSvc1" jndiName="concurrent/${id}">
 <jeeMetadataContext/>
</contextService>
```

- Thread context service with `classloaderContext` and `securityContext`:

```
<contextService jndiName="concurrent/threadContextSvc2">
 <classloaderContext/>
 <securityContext/>
</securityContext/>
```

- Thread context service that inherits `jeeMetadataContext` from `threadContextSvc1` and adds `securityContext`:

```
<contextService jndiName="concurrent/threadContextSvc3"
baseContextRef="threadContextSvc1">
 <securityContext>
</contextService>
```

Example that looks up the default context service:

```
ContextService threadContextSvc =
 (ContextService) new InitialContext().lookup(
 "java:comp/DefaultContextService");
myContextualAsyncCallback = threadContextSvc.createContextualProxy(
 myAsyncCallback, MyAsyncCallback.class);
doSomethingAsync(arg1, arg2, myContextualAsyncCallback);
```

Examples to inject thread context service instances into application components (by using `@Resource`) or look up with resource environment references (`resource-env-ref`).

- Example that uses `@Resource`:

```
@Resource(lookup="concurrent/threadContextSvc1")
ContextService threadContextSvc1;
```

...

```
Callable<Integer> processSalesOrderCompletion = new Callable<Integer>() {
 public Integer call() throws Exception {
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ...update various database tables
 return isSuccessful;
 }
};
```

```
// capture thread context of current application component
execProps = Collections.singletonMap(ManagedTask.TRANSACTION,
ManagedTask.USE_TRANSACTION_OF_EXECUTION_THREAD);
processSalesOrderCompletion = (Callable<Boolean>)
 threadContextSvc1.createContextualProxy(processSaleCompletion, execProps,
Callable.class);
```

```
//later from a different application component
tran.begin();
```

```
...
successful = processSalesOrderCompletion.call();
if (successful)
 tran.commit();
else
 tran.rollback();
```

- Example that specifies `resource-env-ref` in the `web.xml` file:



```

<resource-env-ref>
 <resource-env-ref-name>concurrent/threadContextSvc3</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ContextService</resource-
env-ref-type>
</resource-env-ref>

```

- Example lookup that uses the resource environment reference:

```

ContextService threadContextSvc3 =
(ContextService) new InitialContext().lookup("java:comp/env/concurrent/threadContextSvc3");
Runnable updateAndGetNextFromDatabase = threadContextSvc3.createContextualProxy
(new Runnable() {
 public void run() {
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ... update the database and get next item to process
 }
}, Runnable.class);
barrier = new CyclicBarrier(3, updateAndGetNextFromDatabase);
...

```

## Configuring managed scheduled executors

8.5.5.4

You can configure `ManagedScheduledExecutorService` instances to schedule asynchronous tasks to run with the thread context of the thread from which the task is scheduled. It is a best practice for Java EE applications to avoid directly managing their own threads; therefore, the `ManagedScheduledExecutorService` extends the JSE `ExecutorService` to provide a way to schedule asynchronous tasks within an application server environment. You might also configure the `ManagedScheduledExecutorService` to capture a thread context that is relevant to Java EE applications and propagate it to the thread of the scheduled task.

### About this task

**Important:** In Liberty, managed scheduled executors do not have their own thread pools. Tasks submitted to managed scheduled executor instances run on the common Liberty executor thread pool.

The managed scheduled executor `<concurrent-1.0>` feature is enabled in the `server.xml` file as follows:

```

<featureManager>
 <feature>concurrent-1.0</feature>
</featureManager>

```

Thread context capture and propagation is managed by the context service. A default instance of the context service (`DefaultContextService`) is created by the server and configured to propagate at least `classloaderContext`, `jeeMetadataContext` and `securityContext`. This default context service instance is used if a `ManagedScheduledExecutorService` is created without referring to a specific context service instance or configuring a context service instance directly within. For more information about context service instances, refer to the [Configuring thread context service instances](#) topic.

A default managed scheduled executor instance (`DefaultManagedScheduledExecutorService`) is available as `java:comp/DefaultManagedScheduledExecutorService` and uses the default context service instance for thread context capture and propagation.

### Procedure

Example configuration in the `server.xml` file:

- Managed scheduled executor that is registered in JNDI with the name `concurrent/scheduledExecutor`, and that uses the default context service instance:

```

<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor"/>

```

- Managed scheduled executor with context service configured to capture and propagate `ClassLoaderContext` only:

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor1">
 <contextService>
 <classLoaderContext/>
 </contextService>
</managedScheduledExecutorService>
```

- Managed scheduled executor with `jeeMetadataContext` and `securityContext`:

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor2">
 <contextService>
 <jeeMetadataContext/>
 <securityContext/>
 </contextService>
</managedScheduledExecutorService>
```

- Thread context service that is shared by multiple managed scheduled executors:

```
<contextService id="contextSvc1">
 <jeeMetadataContext/>
</contextService>
```

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor3"
contextServiceRef="contextSvc1"/>
```

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor4" contextServiceRef="contextSvc1"/>
```

## Example

Inject managed scheduled executors into application components (by using `@Resource`) or look up with resource environment references (`resource-env-ref`). Regardless of how the instance is obtained, it can be used interchangeably as `javax.enterprise.concurrent.ManagedScheduledExecutorService` or any of the following superclasses: `java.util.concurrent.ScheduledExecutorService`, `java.util.concurrent.ExecutorService`, `javax.enterprise.concurrent.ManagedExecutorService`

- Example that looks up the default managed scheduled executor:

```
ManagedScheduledExecutorService executor =
 (ManagedScheduledExecutorService) new InitialContext().lookup(
 "java:comp/DefaultManagedScheduledExecutorService");
executor.schedule(beginSalePrices, 12, TimeUnit.HOURS);
executor.schedule(restoreNormalPrices, 60, TimeUnit.HOURS);
```

- Example that uses `@Resource` to inject as `java.util.concurrent.ScheduledExecutorService`:

```
@Resource(lookup="concurrent/scheduledExecutor2")
ScheduledExecutorService executor;
...

// schedule a task to run every half hour from now
Runnable updateSalesReport = new Runnable() {
 public void run() throws Exception {
 // java:comp lookup is possible because <jeeMetadataContext> is configured
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ... query and update various database tables
 }
};
ScheduledFuture<?> future = executor.scheduleAtFixedRate(updateSalesReport, 0, 30, TimeUnit.MINUTES);
```

- Example that uses `@Resource` to inject as `javax.enterprise.concurrent.ManagedScheduledExecutorService`:

```
@Resource(lookup="concurrent/scheduledExecutor2")
ManagedScheduledExecutorService executor;
```

... usage is same as previous example

- Example `<resource-env-ref>` for `java.util.concurrent.ScheduledExecutorService` in the `web.xml` file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/scheduledExecutor1</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ScheduledExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- Example `<resource-env-ref>` for `javax.enterprise.concurrent.ManagedScheduledExecutorService` in the `web.xml` file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/scheduledExecutor2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedScheduledExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- Example lookup that uses a resource environment reference:

```
ManagedScheduledExecutorService executor =
 (ManagedScheduledExecutorService) new InitialContext().lookup("java:comp/env/concurrent/scheduledExecutor2");
executor.schedule(payrollTask, fridaysAtMidnightTrigger);
```

## Configuring managed thread factories

8.5.5.4

You can configure `ManagedThreadFactory` instances to create new threads that run with a thread context of the thread from which the managed thread factory is looked up or injected. It is a best practice for Java EE applications to avoid directly managing their own threads; therefore, the `ManagedThreadFactory` extends the JSE `ThreadFactory` to provide a way to create managed threads within an application server environment. You might also configure the `ManagedThreadFactory` to capture a thread context that is relevant to Java EE applications and propagate it to the new thread.

### About this task

The managed thread factory is available under the `<concurrent-1.0>` feature and enabled in the `server.xml` file as follows:

```
<featureManager>
 <feature>concurrent-1.0</feature>
</featureManager>
```

Thread context capture and propagation is managed by the context service. A default instance of the context service (`DefaultContextService`) is created by the server and configured to propagate at least `classloaderContext`, `jeeMetadataContext` and `securityContext`. This default context service instance is used if a `ManagedThreadFactory` does not specify a context service. For more information about context service instances, refer to the [Configuring thread context service instances](#) topic.

A default instance of `ManagedThreadFactory` (`DefaultManagedThreadFactory`) is available as `java:comp/DefaultManagedThreadFactory` and uses the default context service instance for thread context capture and propagation.

### Procedure

Example configuration in the `server.xml` file:

- Managed thread factory that is registered in JNDI with the name `concurrent/threadFactory`, and that uses the default context service instance:

```
<managedThreadFactory jndiName="concurrent/threadFactory" maxPriority="5"/>
```

- Managed thread factory with context service configured to capture and propagate `securityContext` only:

```
<managedThreadFactory jndiName="concurrent/threadFactory1">
 <contextService>
 <securityContext/>
 </contextService>
</managedThreadFactory>
```

- Managed thread factory with classloaderContext and jeeMetadataContext:

```
<managedThreadFactory jndiName="concurrent/threadFactory2">
 <contextService>
 <classloaderContext/>
 <jeeMetadataContext/>
 </contextService>
</managedThreadFactory>
```

- Thread context service that is shared by multiple managed thread factories:

```
<contextService id="contextSvc1">
 <jeeMetadataContext/>
</contextService>
```

```
<managedThreadFactory jndiName="concurrent/threadFactory3"
contextServiceRef="contextSvc1"/>
```

```
<managedThreadFactory jndiName="concurrent/threadFactory4"
contextServiceRef="contextSvc1"/>
```

## Example

Managed thread factories can be injected into application components (by using @Resource) or looked up with resource environment references (resource-env-ref). Regardless of how the instance is obtained, it can be used interchangeably as javax.enterprise.concurrent.ManagedThreadFactory or java.util.concurrent.ThreadFactory.

- Example that looks up the default managed thread factory:

```
ManagedThreadFactory threadFactory =
 (ManagedThreadFactory) new InitialContext().lookup(
 "java:comp/DefaultManagedThreadFactory");
// Create an executor that always runs tasks with the thread context of the managed thread factory
ExecutorService executor = new ThreadPoolExecutor(
 coreThreads, maxThreads, keepAliveTime, TimeUnit.MINUTES,
 new ArrayBlockingQueue<Runnable>(workRequestQueueSize),
 threadFactory, new ThreadPoolExecutor.AbortPolicy());
```

- Example that uses @Resource to inject as java.util.concurrent.ThreadFactory::

```
@Resource(lookup="concurrent/threadFactory2")
ThreadFactory threadFactory
...

// create a new thread
Thread dailySalesAnalysisTask = threadFactory.newThread(new Runnable() {
 public void run() {
 // java:comp lookup is possible because <jeeMetadataContext> is configured
 DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
 ... analyze the data
 }
});
dailySalesAnalysisTask.start();
```

- Example that uses @Resource to inject as javax.enterprise.concurrent.ManagedThreadFactory:

```
@Resource(lookup="concurrent/threadFactory2")
ManagedThreadFactory threadFactory;
```

... usage is same as previous example

- Example <resource-env-ref> for java.util.concurrent.ThreadFactory in the web.xml file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/threadFactory1</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ThreadFactory</resource-env-ref-type>
</resource-env-ref>
```

- Example `<resource-env-ref>` for `javax.enterprise.concurrent.ManagedThreadFactory` in the `web.xml` file:

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/threadFactory2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedThreadFactory</resource-
env-ref-type>
</resource-env-ref>
```

- Example lookup that uses a resource environment reference:

```
ManagedThreadFactory threadFactory =
 (ManagedThreadFactory) new InitialContext().lookup("java:comp/env/concurrent/threadFactory");
// Create a scheduled executor that always runs tasks with the thread context of the managed thread factory
ScheduledExecutorService executor = Executors.newScheduledThreadPool(5, threadFactory);
... use executor to schedule tasks from any thread
```

## Connecting to Liberty by using JMX

Use this information to access Java Management Extensions (JMX) connectors on Liberty. You can also access the secured JMX connector remotely by using SSL.

### About this task

There are two JMX connectors supported on Liberty, each connector is enabled through a different Liberty feature: `localConnector-1.0` and `restConnector-1.0`.

- The local connector is enabled through the Liberty feature `localConnector-1.0`. Access through the local connector is protected by the policy implemented by the SDK in use. Currently the SDKs require that the client runs on the same host as Liberty, and under the same user ID.
- The REST connector is enabled through the Liberty feature `restConnector-1.0`. Remote access through the REST connector is protected by a single administrator role. In addition, SSL is required to keep the communication confidential. The `restConnector-1.0` feature already includes the `ssl-1.0` feature.

An application deployed on Liberty has unrestricted access to its MBeanServer directory.

A JMX connection to the collective controller enables JMX access to multiple Liberty servers through the same connection. See “Setting up the server-management environment for Liberty by using collectives” on page 907 for more information.

**Restriction:** Do not use JDK options that start with `com.sun.management.jmxremote`, which are described in <http://docs.oracle.com/javase/7/docs/technotes/guides/management/agent.html>, with the Liberty JMX support. Those JDK options adversely affect the Liberty MBean registration framework.

### Procedure

- Connect to the local JMX connector
- Connect to the REST connector
- Work with JMX MBeans

### Configuring local JMX connection to Liberty

You can access the local Java Management Extensions (JMX) connector on Liberty. The local connector is enabled through the Liberty feature `localConnector-1.0`.

### About this task

The local connector is enabled through the Liberty feature `localConnector-1.0`. Access through the local connector is protected by the policy implemented by the SDK in use. Currently the SDKs require that the client runs on the same host as Liberty, and under the same user ID.

**Note:** An application deployed on Liberty has unrestricted access to its MBeanServer directory.

The following section describes how to configure and access the local connector on Liberty.

## Procedure

1. Enable the local connector by using the following code in the `server.xml` file.

```
<featureManager>
 <feature>localConnector-1.0</feature>
</featureManager>
```
2. Access the local connector by using the JConsole tool or JMX client that is installed on the same host.
  - For the JConsole tool, select the local process `ws-server.jar defaultServer` from the connection panel then click **Connect**.
  - For the JMX client, see “Working with JMX MBeans on Liberty” on page 1025.

## Configuring secure JMX connection to Liberty

You can access the secured Java Management Extensions (JMX) connectors on Liberty by using SSL. The secured JMX connection is enabled by the Liberty feature `restConnector-1.0`.

## About this task

The REST connector is enabled through the Liberty feature `restConnector-1.0`. Remote access through the REST connector is protected by a single administrator role. In addition, SSL is required to keep the communication confidential. The `restConnector-1.0` feature already includes the `ssl-1.0` feature.

**Note:** An application deployed on Liberty has unrestricted access to its `MBeanServer` directory.

The following section describes how to configure and access the REST connector on Liberty.

## Procedure

1. Enable the REST connector using the following code in the `server.xml` file.

```
<featureManager>
 <feature>restConnector-1.0</feature>
</featureManager>
```
2. Configure SSL certificates in the `server.xml` file.

Ensure that the CN value of the certificate's subjectDN is the host name of the machine where the server is running, and that the truststore contains the certificate of the server in the jConsole connection.
3. Configure a user or group to the administrator role in the `server.xml` file.
  - Map to the administrator role for Liberty
4. Access the REST connector.

**8.5.5.4** You can access a Liberty REST connector from a Java client or directly through an HTTPS call. A Java client uses the client-side of the connector, which is in `wlp/clients/restConnector.jar` and implements the `javax.management.MBeanServerConnection` interface. HTTPS calls use the server-side of the connector. As to HTTPS calls on the server-side, any programming language that can make HTTPS calls, such as C++, JavaScript, curl, Ruby, and Perl, can use the REST APIs. The REST APIs contain endpoints for management (JMX), file transfer, collective routing and collective deployment.

  - Access the REST connector from a JMX client application or by using the jConsole tool provided in the Java SDK. Use `-J` flags to pass the system properties as Java options and set the class path to include the connector class files. The connector class files are packed in the `clients/restConnector.jar` file.
    - Use the following properties for SSL certificates:

```
-J-Djavax.net.ssl.trustStore=<location of your client trust store>
-J-Djavax.net.ssl.trustStorePassword=<password for the trust store>
-J-Djavax.net.ssl.trustStoreType=<type of trust store>
```

The following example shows the jConsole tool being used with SSL configurations:

```
jconsole -J-Djava.class.path=%JAVA_HOME%/lib/jconsole.jar;
 %JAVA_HOME%/lib/tools.jar;
 %WLP_HOME%/clients/restConnector.jar
-J-Djavax.net.ssl.trustStore=key.jks
-J-Djavax.net.ssl.trustStorePassword=Liberty
-J-Djavax.net.ssl.trustStoreType=jks
```

After the jConsole starts, select **Remote Process**, and enter the JMX service URL:  
service:jmx:rest://<host>:<port>/IBMJMXConnectorREST. The port number is the HTTPS port.  
You must also provide the user name and password.

- **8.5.5.4** Access the REST connector directly using an HTTPS call.

To use HTTPS calls to access REST connectors, you need WebSphere Application Server Liberty 8.5.5.4 or later.

- a. Open a browser at `https://<host>:<port>/IBMJMXConnectorREST/api`, and enter the administrative credentials you specified in step 3.
- b. Examine the available REST APIs. Each item has a description of its behavior, input, output, query parameters, and header.

**Note:** You can specify some JMX REST connection options as system properties. See the Liberty API - WebSphere JMX REST Connector API.

### Mapping the administrator role for Liberty:

You can use `quickStartSecurity` element or any supported user registries for the administrator role mapping in Liberty.

#### About this task

All the JMX methods and MBeans accessed through the REST connector are currently protected by a single role named "administrator". To get started quickly, use `quickStartSecurity` element to configure a single user with administrator role and configure the default SSL configuration.

You can also use any supported user registry. You cannot use `quickStartSecurity` element if you have already configured another user registry. In this case, you have to map users or roles from the registry to the administrator role.

#### Procedure

- Use `quickStartSecurity` element for a single user mapping.

Here is an example showing the minimal required configuration:

```
<featureManager>
 <feature>restConnector-1.0</feature>
</featureManager>
<quickStartSecurity userName="bob" userPassword="bobpassword" />
<keyStore id="defaultKeyStore" password="keystorePassword"/>
```

- Or use the basic registry for administrator role mapping.

Here is an example of the basic registry that gives the user "bob" or the group "group1" administrator role:

```
<basicRegistry>
 <user name="bob" password="bobpassword"/>
 <user name="joe" password="joepassword"/>
 <group name="group1" ...>
 </group>
</basicRegistry>
```

```

<administrator-role>
 <user>bob</user>
 <group>group1</group>
</administrator-role>

```

- Or use the LDAP registry for administrator role mapping (you will need to add the ldapRegistry-3.0 feature to your server.xml file).

Here is an example of the LDAP registry that gives the user "bob" administrator role.

```

<ldapRegistry id="basic" host="" port="">
 <tds.properties ... />
</ldapRegistry>

<administrator-role>
 <user>cn=bob,o=ibm,c=us</user>
</administrator-role>

```

### Developing a JMX Java client for Liberty:

You can develop a Java Management Extensions (JMX) client application to access the secured REST connector of the Liberty server.

#### About this task

Using a JMX remote client application, you can administer the Liberty server through JMX programming.

#### Procedure

- Develop a sample JMX client as follows. The REST connector supports the standard JMX API.

```

import javax.management.remote.JMXServiceURL;
import javax.management.MBeanServerConnection;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import java.util.HashMap;

public class Test {

 public static void main(String[] args) {
 System.setProperty("javax.net.ssl.trustStore", <truststore location>);
 System.setProperty("javax.net.ssl.trustStorePassword", <truststore password>);

 //If the type of the trustStore is not jks, which is default,
 //set the type by using the following line.
 System.setProperty("javax.net.ssl.trustStoreType", <truststore type>);

 try {
 HashMap<String, Object> environment = new HashMap<String, Object>();
 environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
 environment.put(JMXConnector.CREDENTIALS, new String[] { "bob", "bobpassword" });

 JMXServiceURL url = new JMXServiceURL("service:jmx:rest://<host>:<port>/IBMJMXConnectorREST");
 JMXConnector connector = JMXConnectorFactory.newJMXConnector(url, environment);
 connector.connect();
 MBeanServerConnection mbsc = connector.getMBeanServerConnection();
 } catch(Throwable t) {
 ...
 }
 }
}

```

- Optional: Disable host name verification for SSL certificates. The certificates that are installed with the Liberty profile might not contain the host name of where the server is actually running. If you want to disable host name verification of SSL certificates, you can set the system property



`com.ibm.ws.jmx.connector.client.disableURLHostnameVerification` to true, which disables host name verification for all connections. To disable host name verification on a per-connection basis, pass the property as a new environment when you create the JMX connection:

```
HashMap<String, Object> environment = new HashMap<String, Object>();
environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
environment.put("com.ibm.ws.jmx.connector.client.disableURLHostnameVerification", Boolean.TRUE);
environment.put(JMXConnector.CREDENTIALS, new String[] { "bob", "bobpassword" });
...
```

- Optional: Configure JMX REST connector settings by using the environment Map.

```
...
HashMap<String, Object> environment = new HashMap<String, Object>();
environment.put("com.ibm.ws.jmx.connector.client.rest.maxServerWaitTime", 0);
environment.put("com.ibm.ws.jmx.connector.client.rest.notificationDeliveryInterval", 65000);
...
```

- Optional: The Liberty REST connector allows you to specify a custom SSL socket factory that can be used to obtain sockets. If the `javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.j: PKIX path building failed: java.security.cert.CertPathBuilderException: unable to find valid certification path to requested target` exception is displayed, you can create your own `SSLContext` from your own `KeyStores` and then use the `SocketFactory` from that context with the REST connector.

```
KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
InputStream inputStream = new FileInputStream("myTrustStore.jks");
trustStore.load(inputStream, "password".toCharArray());
TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
trustManagerFactory.init(trustStore);
TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();
SSLContext sslContext = SSLContext.getInstance("SSL");
sslContext.init(null, trustManagers, null);
```

```
Map<String, Object> environment = new HashMap<String, Object>();
environment.put(ConnectorSettings.CUSTOM_SSLSOCKETFACTORY, sslContext.getSocketFactory());
environment.put(ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION, true);
environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
environment.put(JMXConnector.CREDENTIALS, new String[] { "admin", "password" });
JMXServiceURL url = new JMXServiceURL("REST", "myhost", 9443, "/IBMJMXConnectorREST");
jmxConn = JMXConnectorFactory.connect(url, environment);
```

## Working with JMX MBeans on Liberty

You can access the attributes and call the operations of Java Management Extensions (JMX) management beans (MBeans) on Liberty. In addition, you can register your own MBeans from an application running on Liberty.

### About this task

The primary interfaces for interacting with MBeans on Liberty are as follows:

- `javax.management.MBeanServer`, which is for application code running on Liberty.
- `javax.management.MBeanServerConnection`, which is for external code running in a separate Java virtual machine.

You can use an instance of either of these interfaces to access the attributes and call the operations of MBeans.

### Procedure

- For application code running on Liberty, you can use a `javax.management.MBeanServer` instance by using the following code:

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
```

...

```
MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
```

...

- For external code running in a separate Java virtual machine, you can use a `javax.management.MBeanServerConnection` instance. See “Developing a JMX Java client for Liberty” on page 1024.

### List of provided MBeans:

Liberty provides a list of MBeans and corresponding management interfaces that you can use to manipulate and monitor the server.

For each MBean or MXBean in the list:

- The name is the `javax.management.ObjectName` value that uniquely identifies the MBean or MXBean. When there are multiple instances of an MBean or MXBean, the `ObjectName` value can contain a wildcard (\*), which is described in the **Comments** entries in this topic.
- The **Management interface** entries specify the name of the Java interface that can be used to construct a proxy object for the MBean or MXBean as described in “Examples of accessing MBean attributes and operations” on page 1029. For more information about the management interface, see the Java API document for Liberty. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

#### WebSphere: feature=channel fw, type=endpoint, name=\*

- **Management interface:** `com.ibm.websphere.endpoint.EndPointInfoMBean`
- **Comments:** One instance is available for each endpoint in the system, where \* is a unique endpoint name.

#### WebSphere: feature=restConnector, type=FileService, name=FileService

- **Management interface:** `com.ibm.websphere.filetransfer.FileServiceMXBean`
- **Comments:** This MXBean enables you to perform various file-related operations on the host where Liberty resides.

You can find its class and API documentation in the following locations:

```
liberty_home/dev/api/ibm/com.ibm.websphere.appserver.api.restConnector_version.jar
```

```
liberty_home/dev/api/ibm/javadoc/com.ibm.websphere.appserver.api.restConnector_version-javadoc.zip
```

The exposed operations include the ability to query certain metadata (last modified date, size, and so on) for a given file or directory and also to query all child files (and corresponding metadata) for a given directory. Support for archive creation and expansion is also provided, which can be useful to compress Liberty log files or to extract an application before deploying it.

This MXBean contains two attributes: the read list and the write list. They represent the lists of locations that users can read or write to when using the FileService or FileTransfer capabilities provided by Liberty. Through the MXBean, these attributes can only be read, but they can be configured or customized through the following elements in the `server.xml` file:

```
<remoteFileAccess>
 <readDir>${server.output.dir}/logs</readDir>
 <readDir>${server.output.dir}/apps</readDir>
 <writeDir>${server.output.dir}/dropins</writeDir>
</remoteFileAccess>
```

If the `readDir` element is not specified, the default is the combination of: `${wlp.install.dir}`, `${wlp.user.dir}`, and `${server.output.dir}`. If a `writeDir` element is not specified, the default is the empty set.

The `restConnector-1.0` feature must be included in the `server.xml` file in order for this MBean to be loaded and to honor its configuration elements

Using Liberty-defined variables is allowed with all the server-side parameters that take a string representing a file path. Such variables are defined on the `liberty_home/README.TXT` file.

#### **WebSphere: feature=restConnector, type=FileTransfer, name=FileTransfer**

- **Management interface:** `com.ibm.websphere.filetransfer.FileTransferMBean`
- **Comments:** This MBean allows you to perform various file-transfer operations on the host where Liberty resides.

You can find its class and API documentation in the following locations:

`liberty_home/dev/api/ibm/com.ibm.websphere.appserver.api.restConnector_version.jar`

`liberty_home/dev/api/ibm/javadoc/com.ibm.websphere.appserver.api.restConnector_version-javadoc.zip`

This MBean is registered on the `PlatformMBeanServer` from the same JVM that its corresponding Liberty process is running, but it can be accessed only by using the IBM JMX REST Connector. The connection can be local or remote, but the REST Connector must be used.

The exposed operations include the ability to download, upload, and delete a file. Each read and write request on the server is bound to the configurable read and write lists that are accessed through the `FileServiceMBean`. The `FileTransferMBean` can also be fully accessed and operated from the built-in Java JConsole, provided that the JConsole is connected through the IBM JMX REST Connector.

Using Liberty-defined variables is allowed with all the server-side parameters that take a string representing a file path. Such variables are defined on the `liberty_home/README.TXT` file.

8.5.5.5

#### **WebSphere: name=com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean**

- **Management interface:** `com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean`
- **Comments:** The `ServerXMLConfigurationMBean` provides an interface for retrieving the file paths of all server configuration files known to the server. The MBean is available from the Kernel, so you do not need to enable a special feature. You can find the MBean class and API documentation in the following locations:

– `${wlp.install.dir}/dev/api/ibm/com.ibm.websphere.appserver.api.config_version.jar`

– `${wlp.install.dir}/dev/api/ibm/javadoc/com.ibm.websphere.appserver.api.config_version-javadoc.zip`

8.5.5.4

#### **WebSphere: name=com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean**

- **Management interface:** `com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean`
- **Comments:** The `RuntimeUpdateNotificationMBean` provides notifications for server runtime updates. The user data object attached to the notification is a `java.util.Map`. The notification type for runtime update notifications emitted by this MBean is `com.ibm.websphere.runtime.update.notification`.

8.5.5.5

#### **WebSphere: name=com.ibm.ws.config.mbeans.FeatureListMBean**

- **Management interface:** `com.ibm.websphere.config.mbeans.FeatureListMBean`
- **Comments:** The `FeatureListMBean` exposes a single method to generate an XML report on all the features installed at run time. The MBean is available from the Kernel, so you do not need to enable a special feature. You can find the MBean class and API documentation in the following locations:
  - `${wlp.install.dir}/dev/api/ibm/com.ibm.websphere.appserver.api.config_version.jar`
  - `${wlp.install.dir}/dev/api/ibm/javadoc/com.ibm.websphere.appserver.api.config_version-javadoc.zip`

8.5.5.5

**WebSphere:name=com.ibm.ws.config.serverSchemaGenerator**

- **Management interface:**com.ibm.websphere.config.mbeans.ServerSchemaGenerator
- **Comments:** The ServerSchemaGenerator MBean exposes methods to generate schema from the installed image, the most used way, or from a current runtime. The MBean is available from the Kernel, so you do not need to enable a special feature. You can find the MBean class and API documentation in the following locations:
  - `${wlp.install.dir}/dev/api/ibm/com.ibm.websphere.appserver.api.config_version.jar`
  - `${wlp.install.dir}/dev/api/ibm/javadoc/com.ibm.websphere.appserver.api.config_version-javadoc.zip`

**WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig**

- **Management interface:**com.ibm.websphere.webcontainer.GeneratePluginConfigMBean
- **Comments:** See “Configuring a web server plug-in for Liberty” on page 985.

**WebSphere:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean**

- **Management interface:**com.ibm.websphere.filemonitor.FileNotificationMBean

**WebSphere:service=com.ibm.websphere.application.ApplicationMBean,name=\***

- **Management interface:**com.ibm.websphere.application.ApplicationMBean
- **Comments:** One instance is available for each application in the system, where \* is a unique application name.

8.5.5.6

**WebSphere:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,\***

- **Management interface:**com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean
- **Comments:** One instance is available for each Connection Manager in the system, including those created in the following contexts:
  - When explicitly configured in the server configuration
  - When implicitly created because of `@DataSourceDefinition` or `@ConnectionFactoryDefinition` annotations
  - When created as a result of a connection factory or data source in the server configuration

The mbean instance is not available until the corresponding connection factory or data source is first used.

To narrow the connection manager instance, you can specify additional attributes, such as those shown in the following examples:

```
WebSphere:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,jndiName=jdbc/db2,*
```

```
WebSphere:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=jmsConnectionFactory[cf1]/connectionManager[default]
```

```
WebSphere:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=databaseStore[dbstore1]/dataSource[default-0]/connectionManager[default]
```

```
WebSphere:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,jndiName=java.module/env/jdbc/ds3,application=MyApp,module=...
```

**WebSphere:type=JvmStats**

- **Management interface:** com.ibm.websphere.monitor.jmx.JvmMXBean
- **Comments:** Available when the monitor-1.0 feature is enabled. See “JVM monitoring” on page 1423.

**WebSphere:type=ServletStats,name=\***

- **Management interface:**com.ibm.websphere.webcontainer.ServletStatsMXBean
- **Comments:** When the monitor-1.0 feature is enabled, one instance is available for each servlet that has been served, where \* is of the form `<AppName>.<ServletName>`. See “Web application monitoring” on page 1424.

### WebSphere:type=ThreadPoolStats,name=Default Executor

- **Management interface:** com.ibm.websphere.monitor.jmx.ThreadPoolMXBean
- **Comments:** Available when the monitor-1.0 feature is enabled. See “ThreadPool monitoring” on page 1425.

8.5.5.3

### WebSphere:feature=kernel,name=ServerInfo

- **Management interface:** com.ibm.websphere.kernel.server.ServerInfoMBean
- **Comments:** The ServerInfoMbean interface is used to retrieve information about the running server. Search the following directories for the class and API documentation:

*liberty\_home/dev/api/ibm/com.ibm.websphere.appserver.api.kernel.service\_version-javadoc.zip*

*liberty\_home/dev/api/ibm/com.ibm.websphere.appserver.api.kernel.service\_version.jar*

The exposed operations include a method to retrieve the product install and user directory locations, the default host name, the server name, the product version, the Java specification version, and the Java Runtime version.

### Examples of accessing MBean attributes and operations:

You can use Liberty to access the attributes, and call the operations, of Java Management Extensions (JMX) management beans (MBeans).

After you obtain an MBeanServer instance (for an application running on Liberty) or an MBeanServerConnection instance (for an external client), you can access the attributes or call the operations of MBeans provided by Liberty. See “Working with JMX MBeans on Liberty” on page 1025.

The following code examples assume the variable `mbs` is an `MBeanServer` or `MBeanServerConnection` instance. You can use the provided methods to access the attributes and operations in a similar way to Java reflection. Alternatively, each MBean has a management interface with getter methods for the attributes and methods for the operations. You can use these interfaces by implementing one of the `javax.management.jmx.newMBeanProxy` methods or one of the `javax.management.jmx.newMXBeanProxy` methods for MXBeans to obtain a proxy object. The name of a management interface ends with “MXBean”. For the names of the management interfaces, see “List of provided MBeans” on page 1026.

#### Example 1: Check the state of application "myApp"

```
import javax.management.ObjectName;
import javax.management.JMX;
import com.ibm.websphere.application.ApplicationMBean;
...

ObjectName myAppMBean = new ObjectName(
"WebSphere:service=com.ibm.websphere.application.ApplicationMBean,name=myApp");
if (mbs.isRegistered(myAppMBean)) {
 String state = (String) mbs.getAttribute(myAppMBean, "State");
 // alternatively, obtain a proxy object
 ApplicationMBean app = JMX.newMBeanProxy(mbs, myAppMBean, ApplicationMBean.class);
 state = app.getState();
}
```

#### Example 2: Get response time statistics for servlet "Example Servlet" from application "myApp"

```
import javax.management.ObjectName;
import javax.management.openmbean.CompositeData;
import javax.management.JMX;
import com.ibm.websphere.webcontainer.ServletStatsMXBean;
...

ObjectName servletMBean = new ObjectName("WebSphere:type=ServletStats,name=myApp.Example Servlet");
```

```

if (mbs.isRegistered(servletMBean)) {
 CompositeData responseTimeDetails = (CompositeData) mbs.getAttribute(servletMBean, "ResponseTimeDetails");
 CompositeData responseTimeReading = (CompositeData) responseTimeDetails.get("reading");
 Double mean = (Double) responseTimeReading.get("mean");
 Double standardDeviation = (Double) responseTimeReading.get("standardDeviation");
 // alternatively, obtain a proxy object
 ServletStatsMXBean servletStats = JMX.newMXBeanProxy(mbs, servletMBean, ServletStatsMXBean.class);
 StatisticsMeter meter = servletStats.getResponseTimeDetails();
 StatisticsReading reading = meter.getReading();
 mean = reading.getMean();
 standardDeviation = reading.getStandardDeviation();
}

```

### Example 3: Create a web server plug-in configuration file

```

import com.ibm.websphere.webcontainer.GeneratePluginConfigMBean;

...

ObjectName pluginMBean = new ObjectName("WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig");
if (mbs.isRegistered(pluginMBean)) {
 mbs.invoke(pluginMBean, "generatePluginConfig", new Object[] {
 "installRoot", "serverName", new String[] {
 String.class.getName(), String.class.getName()
 });
 // alternatively, use a proxy object
 GeneratePluginConfigMBean plugin = JMX.newMBeanProxy(mbs, name, GeneratePluginConfigMBean.class);
 plugin.generatePluginConfig("installRoot", "serverName");
}

```

### Example 4: Query status of Web Service Endpoint

```

import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;

...

// Init mbs as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-Bus,
 type=Bus.Service.Endpoint,service=\"{http://jaxws.samples.ibm.com:jmx}/TestEndpointService\",
 port=\"TestEndpoint\",instance.id=1816106538");
MBeanInfo beanInfo = mbs.getMBeanInfo(objName);

// Go through attributes to find the interested one
for (MBeanAttributeInfo attr : beanInfo.getAttributes()) {
 if (attr.getName().equals("State")) {
 String status = String.valueOf(mbs.getAttribute(objName, attr.getName()));
 break;
 }
}

```

### Example 5: Shut down the CXF server bus

```

import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;

...

// Init mbsc as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-Bus,
 type=Bus,instance.id=1618108530");
MBeanInfo beanInfo = mbs.getMBeanInfo(objName);

// Go through operation to find the interested one and invoke
for (MBeanOperationInfo operation : beanInfo.getOperations()) {
 if (operation.getName().equals("shutdown")) {

```

```

 mbs.invoke(objName, operation.getName(), new Object[] { true }, new String[] { boolean.class.getName() });
 break;
}
}

```

### Examples of registering MBeans:

An application can register its own MBean instances on Liberty. That MBean instance can then be used by other applications or external administrators.

Any application can register an MBean by using an MBeanServer instance. Suppose an application contains a class called `org.example.Example` that implements the interface `org.example.ExampleMBean`, which defines some attributes and operations. As in the following example, the application might simply instantiate the `Example` class then register it using a unique `ObjectName`. If the `ObjectName` chosen is already in use, a `javax.management.InstanceAlreadyExistsException` is reported.

```

import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.Example;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
Object mbean = new Example();
ObjectName name = new ObjectName("org.example.MyApplication:name=Example");
mbs.registerMBean(mbean, name);

```

In addition, an application might register an MBean that extends `java.lang.ClassLoader` and provides access to any number of MBean implementation classes. Then you can use any other JMX client, local or remote, to create and register MBeans provided by the application. For example, suppose the application has an MBean class `org.example.ApplicationClassLoader` that performs the following tasks:

- Implements any empty interface `org.example.ApplicationClassLoaderMBean`
- Extends `java.lang.ClassLoader`, and
- Provides access to the `org.example.Example` MBean implementation class

The application can register an instance of `ApplicationClassLoader` to make the `Example` MBean available to other JMX clients as follows:

```

import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.ApplicationClassLoader;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
Object classLoader = new ApplicationClassLoader();
ObjectName name = new ObjectName("org.example.MyApplication:name=ClassLoader");
mbs.registerMBean(classLoader, name);

```

Any JMX client can create an `Example` instance. The following example assumes the variable `mbs` is an `MBeanServer` or `MBeanServerConnection` instance. See “Working with JMX MBeans on Liberty” on page 1025.

```

import javax.management.ObjectName;

...

ObjectName loaderName = new ObjectName("org.example.MyApplication:name=ClassLoader");
ObjectName exampleName = new ObjectName("org.example.MyApplication:name=Example");
mbs.createMBean("org.example.Example", exampleName, loaderName);

```

If necessary, you can use other forms of the `MBeanServer.createMBean` method to create the MBean by using non-default constructors.

For more information about the management interface, see the Java API document for Liberty. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

### Example of setting up a JMX routing environment:

You can use Liberty to call Java Management Extensions (JMX) management beans (MBeans) on a collective member server through a collective controller server.

**Note:** **Distributed operating systems** **IBM i** The `collectiveController-1.0` feature and its capabilities are available only in WebSphere Application Server Liberty Network Deployment and WebSphere Application Server Liberty for z/OS. The feature is not available in WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core. If you have a WebSphere Application Server Liberty Network Deployment installation, you can use its `collectiveController-1.0` feature to work with collective members from WebSphere Application Server Liberty, WebSphere Application Server Liberty - Express, or WebSphere Application Server Liberty Core installations.

The `collectiveMember-1.0` feature enables a server to be managed by a collective controller (the `collectiveController-1.0` feature). After a server is configured to be managed by a collective controller, you can directly call any MBeans on the collective member through the collective controller server.

The following is an example of how to call MBeans on a collective member through a collective controller server.

```
// Set up the trust store to the collective controller server.

System.setProperty("javax.net.ssl.trustStore", "<trustStore for https connection to collective controller>");
System.setProperty("javax.net.ssl.trustStorePassword", "<trustStore password>");

Map<String, Object> environment = new HashMap<String, Object>();
environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
environment.put(JMXConnector.CREDENTIALS, new String[] { "<username>", "<password>" });
environment.put(ClientProvider.DISABLE_HOSTNAME_VERIFICATION, true);
environment.put(ClientProvider.READ_TIMEOUT, 2 * 60 * 1000);

JMXServiceURL url = new JMXServiceURL(
 "REST", "<hostname of collective controller server>", <https port>, "/IBMJMXConnectorREST");
jmxConnector = JMXConnectorFactory.connect(url, environment);
MBeanServerConnection exmbc = jmxConnector.getMBeanServerConnection();

// You have a MBeanServerConnection now; at this point, however, all of your MBean calls
// are on the collective controller server.

// The next few lines of code are to set up the routing context so that all calls
// can be routed to a collective member.

ObjectName rmObjectName = new ObjectName(
 "WebSphere:feature=collectiveController,type=RoutingContext,name=RoutingContext");

// Call the MBeanRoutingContext MBean to set up the routing context.

Object rcObj = connection.invoke(rmObjectName, "assignServerContext",
 new Object[] {
 "<hostname of the collective member>", "<collective member server usr dir>", "<collective member server name>"
 },

// With the collective-member server usr dir and collective-member server name,
// the managed server can be uniquely identified on a host.

 new String[] { "java.lang.String", "java.lang.String", "java.lang.String" });

if (rcObj instanceof Boolean) {
 Boolean result = (Boolean) rcObj;
}
```



```

if (result.booleanValue()) {
 System.out.println("routing context is configured correctly");
}
Or if (!result.booleanValue()) {
 System.out.println("routing context result is false");
}
} else {
 System.out.println("failed to configure routing context");
}
}

```

If the routing context is configured correctly, all future calls to this MBeanServerConnection will be routed to target collective member server.

## Establishing a JMX MBean Liberty server connection

You can use Jython-based scripts to establish a Java Management Extensions (JMX) MBean Liberty server connection.

### Before you begin

You must obtain and install the Jython version of your choice before you can perform this procedure. Without a Jython runtime, the instructions will fail.

### Procedure

1. Set up the environment.

The files that you need are located in *liberty\_home/clients/jython*.

- a. Copy the `lib/restConnector.py` file to *jython\_home/Lib*.
- b. Set the classpath for `restConnector.jar` in *liberty\_home/clients*.

```
set CLASSPATH=%CLASSPATH%;c:\wlp\clients\restConnector.jar
```

2. Run the utility.

**Example 1:** Getting a simple connection using `connector.connect(host,port,user,password)`

```

from restConnector import JMXRESTConnector
JMXRESTConnector.trustStore = "c:/key.jks"
JMXRESTConnector.trustStorePassword = "Liberty"

```

```

connector = JMXRESTConnector()
connector.connect("foo.bar.com",9443,"theUser","thePassword")
mconnection = connector.getMBeanServerConnection()
mconnection.invoke(...)
connector.disconnect()

```

**Example 2:** Getting an advanced connection using `connector.connect(host,port,map)` with user-provided properties

```

import java
import javax
import jarray
import com.ibm.websphere.jmx.connector.rest
import com.ibm.ws.jmx.connector.client.rest

map=java.util.HashMap()
map.put("jmx.remote.provider.pkgs","com.ibm.ws.jmx.connector.client")
map.put(javax.management.remote.JMXConnector.CREDENTIALS,jarray.array(["theUser","thePassword"],java.lang.String))
map.put(com.ibm.ws.jmx.connector.client.rest.ClientProvider.READ_TIMEOUT,2*60*1000)
map.put(com.ibm.websphere.jmx.connector.rest.ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION, True)

connector = JMXRESTConnector()
connector.connect("foo.bar.com",9443,map)
mconnection = connector.getMBeanServerConnection()
mconnection.invoke(...)
connector.disconnect()

```

**Example 3:** Registering a notification listener

```

import java
import javax

```

```

from restConnector import JMXRESTConnector
from restConnector import BaseNotificationListener

class SampleNotificationListener(BaseNotificationListener):
 def __init__(self):
 pass

 def handleNotification(self,notification,handback):
 print "Notification received:"
 print " Source: " + notification.getSource().toString()
 print " Type: " + notification.getType()
 print " Message: " + notification.getMessage()

main starts here

JMXRESTConnector.trustStore = "c:/key.jks"
JMXRESTConnector.trustStorePassword = "Liberty"

connector=JMXRESTConnector()
connector.connect("foo.bar.com",9443,"theUser","thePassword")
mconnection=connector.getMBeanServerConnection()

listener=SampleNotificationListener()
handback=java.lang.Object()

notifier1=javax.management.ObjectName("web:name=Notifier1")
mconnection.addNotificationListener(notifier1,listener,None,handback)

```

#### **JMXRESTConnector.trustStore**

Sets the path to where the SSL key file is stored

#### **JMXRESTConnector.trustStorePassword**

Sets the password for the key

#### **JMXRESTConnector.connect(host,port,user,password)**

Creates a connector to the server

#### **JMXRESTConnector.connect(host,port,map)**

Creates a connector with user properties

#### **JMXRESTConnector.getMBeanServerConnection**

Gets a connection to the MBean server

#### **JMXRESTConnector.disconnect()**

Closes the connection

## What to do next

After a connection to the MBean server is established, you can make calls to the MBean server by using the `invoke(...)` method.

**Note:** A library of Jython scripts is available for you to download from the Liberty Repository.

## File transfer

The `restConnector-1.0` feature includes the `FileTransfer` and `FileService` MBeans. The `FileTransfer` MBean supports delete, upload, and download operations to and from a running Liberty server. The `FileService` MBean provides access to directory lists and file metadata, and it also provides archive operations such as **create** and **expand**.

The `FileTransfer` and `FileService` MBeans are useful for carrying out remote operations on a Liberty, such as updating the configuration or installing an application. A configuration update can be performed remotely by uploading an updated `server.xml` file for the target Liberty server. An application can be

installed by uploading both the application archive and an updated `server.xml` file or simply by uploading the application archive to the monitored dropins folder.

The `FileTransfer` MBean includes configurable read and write lists so that you can control the directories that can be read or written when using the `FileTransfer` MBean.

See the section on the `FileTransfer` MBean in “List of provided MBeans” on page 1026 for information on how to configure the `restConnector-1.0` feature and control the `FileTransfer` MBean read and write lists.

The `uploadFile` method from `FileTransfer` contains a boolean called `expandOnCompletion` that allows a user to upload and expand the archive with a single MBean invocation. A directory is created with the same name as the archive on the target path and the `FileService` MBean is automatically invoked to expand the archive.

**Example:** A call to `uploadFile` with parameters `{"C:/temp/myArchive.zip", "${server.output.dir}/myArchive.zip", true}` will result in a `myArchive.zip` directory under `${server.output.dir}` that contains the extracted contents of the archive being created.

## Transferring files in a Liberty collective

A Liberty controller enables special file transfer capabilities within a Liberty collective. The most advanced such functionality is file transfer for multiple hosts within a single REST call. You can use the `FileTransfer` and `FileService` MBeans in a Liberty collective to perform file actions on any Liberty server in the collective. This includes both Liberty servers configured as collective controllers and as collective members.

### About this task

When you establish a remote JMX connection to a collective controller, you can use the `RoutingContext` MBean to direct your `FileTransfer` and `FileService` MBean calls to execute on any Liberty server in the collective. The collective controller takes care of routing the request and creating authorized connections between the collective controller and the target collective member.

By routing file operations to specific collective members, you can perform configuration-file updates and install applications on any Liberty server in the collective.

The `FileTransfer` MBean additionally can perform operations on a host computer in a collective whether or not there is a collective member on that host. By registering the host computer with the Liberty collective and specifying a `RoutingContext` that specifies that host, the `FileTransfer` command can be used to upload or download files to or from that host system. For example, you can upload and expand a Liberty archive to push out Liberty to new hosts.

The upload and extract operation in a routing environment has a more complex logic than that in the non-routing environment:

- If the target host machine has access to an **unzip** command on its path, that command is invoked to extract the archive.
- If an **unzip** command is not found, the process uses a Java-based archive extraction.
  1. A small Liberty-built jar file is temporarily pushed into the target host.
  2. A path to Java is found by checking the following:
    - Configured Java-home value that was setup during host registration
    - Configured `JAVA_HOME` variable visible to RXA
    - Configured `JRE_HOME` variable visible to RXA
    - Java home found on the path
  3. The custom Java jar file is invoked to extract the original archive.

4. The custom Java jar file is deleted from the target host.
5. If the archive being expanded by the custom Java jar file is a Liberty archive, the process recursively sets the permissions of its wlp/bin folder to 755 to allow for remote management of that Liberty instance.

The FileTransfer MBean uses authorization information stored in the collective controller for either the target host computer or collective member. This information was stored when the host computer or collective member was registered. See “Registering host computers with a Liberty collective” on page 921 for more information on setting up this information for the host computer.

File-transfer operations directed at the host computer use the authorization information stored for the host computer. File-transfer operations directed at a collective member use the authorization overridden by the collective member, if any, or use the information stored for the host computer by default. See “Overriding Liberty server host information” on page 919 for more information on overriding host information.

The FileService MBean operates on Liberty (not only on host computers) and uses the authorization configurations from Liberty. It does not use host-computer authorization information.

## Transferring files to and from a collective member or registered host with REST calls

8.5.5.6

You can perform routed file transfer operations from a collective controller to a collective member or to a registered host of a collective by invoking REST APIs. The operations include downloading files from a remote location, uploading files to a remote location, and deleting files in a remote location. Use the **GET**, **POST**, or **DELETE** REST APIs. The file transfer operations occur within an IBM JMX REST Connector.

### Before you begin

To perform file transfer operations from a collective controller to a member server, the server must be joined as a member to the collective. See “Configuring a Liberty collective” on page 912.

To perform file transfer operations to a host computer, the host must be registered with the collective controller. See “Registering host computers with a Liberty collective” on page 921.

### Procedure

1. Optional: To read about the REST APIs, point a browser at `https://controller_host_name:controller_port_name/IBMJMXConnectorREST/api` and enter the controller administrative user ID and password to log in. The controller must be running to view the REST API documentation. Alternatively, you can use a collective member host, port, login user ID and password to view the REST API documentation.

The **File Transfer** and **Routing** sections describe the APIs used to transfer files between the collective controller and a member server or a registered host.

The `collectiveController-1.0` and `collectiveMember-1.0` features enable the `restConnector-1.0` feature, which provides file transfer capability. Thus, collective controllers and members do not need to specify `restConnector-1.0` in a feature manager to view the REST APIs or perform file transfer operations. A stand-alone server configuration might need the `restConnector-1.0` feature to view the REST APIs.

2. Unless you work directly with a Liberty instance, set the routing context as HTTP headers.

- Member server routing

```
com.ibm.websphere.jmx.connector.rest.routing.hostName=string
com.ibm.websphere.jmx.connector.rest.routing.serverName=string
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir=string
```

- Registered host routing

```
com.ibm.websphere.jmx.connector.rest.routing.hostName=string
```

3. Ensure the target file is within the configurable read/write directories of the server for file transfer operations with a member server, or within configurable read/write directories of the host for file transfer operations with a registered host.

4. Invoke REST APIs that download, upload, or delete files.

*{filePath}* must be URL-encoded. For routing operations with registered hosts, *{filePath}* must be an absolute path and cannot contain Liberty variables.

- Download one file from a member server or registered host using the **GET** operation.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/{filePath}
```

- Upload one file to a member server or registered host using the **POST** operation.

```
POST https://controller_host:controller_port/IBMJMXConnectorREST/file/{filePath}
```

- Delete one file from a member server or registered host using the **DELETE** operation.

```
DELETE https://controller_host:controller_port/IBMJMXConnectorREST/file/{filePath}
```

- Delete multiple files from a member server or registered host using the **POST** operation.

```
POST https://controller_host:controller_port/IBMJMXConnectorREST/file/collection
```

## Example

To download the `myFile.txt` file from the member server `myServerA` on the host `myTarget.com` with a user directory of `C:/server/wlp`:

1. Set the member server routing context as HTTP headers.

```
com.ibm.websphere.jmx.connector.rest.routing.hostName=myTarget.com
com.ibm.websphere.jmx.connector.rest.routing.serverName=myServerA
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir=C:/server/wlp
```

2. Invoke a **GET** call to download the file.

```
GET https://myTarget.com:9443/IBMJMXConnectorREST/file/C%3A%2Ftemp%2FmyFile.txt
```

To download the `myFile.txt` file from the registered host `myTarget.com`:

1. Set the registered host routing context as an HTTP header.

```
com.ibm.websphere.jmx.connector.rest.routing.hostName=myTarget.com
```

2. Invoke a **GET** call to download the file.

```
GET https://myTarget.com:9443/IBMJMXConnectorREST/file/C%3A%2Ftemp%2FmyFile.txt
```

To download the `server.xml` file from the member server `myServerA` on the host `myTarget.com` with a user directory of `C:/server/wlp`:

1. Set the member server routing context as HTTP headers.

```
com.ibm.websphere.jmx.connector.rest.routing.hostName=myTarget.com
com.ibm.websphere.jmx.connector.rest.routing.serverName=myServerA
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir=C:/server/wlp
```

2. Invoke a **GET** call to download the `${server.config.dir}/server.xml` file.

```
GET https://myTarget.com:9443/IBMJMXConnectorREST/file/%24{server.config.dir}%2Fserver.xml
```

## What to do next

Get status or details on the REST call. See “Getting status on a REST call for multiple registered hosts” on page 1040.

## Uploading files to multiple registered hosts with a single REST call

8.5.5.6

You can upload files from a collective controller to multiple registered hosts of a collective by invoking the **POST** REST API. The file transfer operations occur within an IBM JMX REST Connector.

## Before you begin

Create a collective controller. See “Configuring a Liberty collective” on page 912.

Register each remote host computer with the collective controller. See “Registering host computers with a Liberty collective” on page 921.

## Procedure

1. Optional: To read about the REST APIs, point a browser at `https://controller_host_name:controller_port_name/IBMJMXConnectorREST/api` and enter the controller administrative user ID and password to log in. The controller must be running to view the REST API documentation.

The **File Transfer** and **Routing** sections describe the APIs used to transfer files between the collective controller and a member server or a registered host.

The `collectiveController-1.0` feature enables the `restConnector-1.0` feature, which provides file transfer capability. Thus, collective controllers do not need to specify `restConnector-1.0` in a feature manager to view the REST APIs or perform file transfer operations. A stand-alone server configuration might need the `restConnector-1.0` feature to view the REST APIs.

2. List the target hosts in an HTTP header for the collective controller.

```
com.ibm.websphere.collective.hostNames=comma-separated_list_of_target_hosts
```

Optionally, set other HTTP headers:

- Specify whether to perform the file uploads and the action asynchronously. Default is `false`. To change the default, specify `true` for *boolean*.

```
com.ibm.websphere.jmx.connector.rest.asyncExecution=boolean
```

- Specify a set of actions to perform after the file uploads. An existing built-in action is `com.ibm.websphere.jmx.connector.rest.postTransferAction.join`, which joins the Liberty servers inside the incoming Liberty archive to the collective. Use the header to list other custom actions. If you list more than one custom action, delimit the action with a URL-encoded comma character.

```
com.ibm.websphere.jmx.connector.rest.postTransferAction=comma-separated_list_of_actions
```

To enable the custom actions, add the following configuration to the collective controller `server.xml`:

```
<hostAccess enableCustomActions="true" />
```

- Specify a list of options to pass into the post-transfer actions. This list of options must be either null or contain the same number of list items as the `postTransferAction` header list, where the item index of each option must match the index of its corresponding action. If you list more than one option, delimit the action with a URL-encoded comma character.

```
com.ibm.websphere.jmx.connector.rest.postTransferAction.options=comma-separated_list_of_options
```

If you specify the built-in action `com.ibm.websphere.jmx.connector.rest.postTransferAction.join`, the corresponding option must be:

```
--user=adminUser --password=adminPw --keystorePassword=keystorePw [--rpcUser=rpcUser --rpcPassword=rpcPw]
```

- Specify environment variables to set before the transfer actions run. The payload of the header is a JSON object where each JSON key is an environment variable and each JSON value is its corresponding value. If you specify system paths, use forward slashes (/).

```
com.ibm.websphere.jmx.connector.rest.transferEnvVars=list_of_environment_variables
```

3. Upload a file to multiple registered hosts by using the **POST** operation.

- a. Ensure that the target location is within a configurable write directory of the hosts.
- b. Invoke the **POST** operation.

```
POST https://controller_host:controller_port/IBMJMXConnectorREST/file/{filePath}[?expandOnCompletion=boolean&local=boolean]
```

- `{filePath}` is a UTF-8 URL-encoded absolute path that specifies the target location. For example, if the file is `C:/temp/myFile.txt`, the path is `C%3A%2Ftemp%2FmyFile.txt`.
- `expandOnCompletion` is an optional query parameter that toggles an automatic expansion of the uploaded archive. The default value is `false`.

- `local` is an optional query parameter that specifies whether the file to upload is already in the controller. If true, then the payload of this **POST** request is a string that represents the source file location inside the controller. The default value is false.
- If the `local` query parameter is not used, or explicitly set to false, the **POST** payload is the binary content of the source file itself.

## What to do next

Get status or details on the REST call. See “Getting status on a REST call for multiple registered hosts” on page 1040.

## Deleting files from multiple registered hosts with a single REST call

8.5.5.6

You can delete files on multiple registered hosts of a collective from a collective controller by invoking the **DELETE** REST API. The file deletions occur within an IBM JMX REST Connector.

### Before you begin

Create a collective controller. See “Configuring a Liberty collective” on page 912.

Register each remote host computer with the collective controller. See “Registering host computers with a Liberty collective” on page 921.

### Procedure

1. Optional: To read about the REST APIs, point a browser at `https://controller_host_name:controller_port_name/IBMJMXConnectorREST/api` and enter the controller administrative user ID and password to log in. The controller must be running to view the REST API documentation.

The **File Transfer** and **Routing** sections describe the APIs used to transfer files between the collective controller and a member server or a registered host.

The `collectiveController-1.0` feature enables the `restConnector-1.0` feature, which provides file transfer capability. Thus, collective controllers do not need to specify `restConnector-1.0` in a feature manager to view the REST APIs or perform file transfer operations. A stand-alone server configuration might need the `restConnector-1.0` feature to view the REST APIs.

2. List the target hosts in an HTTP header for the collective controller.

```
com.ibm.websphere.collective.hostNames=comma-separated_list_of_target_hosts
```

Optionally, set other HTTP headers:

- Specify whether to perform the file deletion and the action asynchronously. Default is false. To change the default, specify true for *boolean*.

```
com.ibm.websphere.jmx.connector.rest.asyncExecution=boolean
```

- Specify a set of actions to perform before the file deletion. An existing built-in action is `com.ibm.websphere.jmx.connector.rest.preTransferAction.remove`, which removes the Liberty servers inside the Liberty directory to be deleted from the collection. Use the header to list other custom actions. If you list more than one custom action, delimit the action with a URL-encoded comma character.

```
com.ibm.websphere.jmx.connector.rest.preTransferAction=comma-separated_list_of_actions
```

To enable the custom actions, add the following configuration to the collective controller `server.xml`:

```
<hostAccess enableCustomActions="true" />
```

- Specify a list of options to pass into the pre-transfer actions. This list of options must be either null or contain the same number of list items as the `preTransferAction` header list, where the item

index of each option must match the index of its corresponding action. If you list more than one option, delimit the action with a URL-encoded comma character.

```
com.ibm.websphere.jmx.connector.rest.preTransferAction.options=comma-separated_list_of_options
```

If you specify the built-in action

```
com.ibm.websphere.jmx.connector.rest.preTransferAction.remove, the corresponding option must be:
```

```
--user=adminUser --password=adminPw --keystorePassword=keystorePw [--rpcUser=rpcUser --rpcPassword=rpcPw]
```

- Specify credentials to use for the delete action. The payload of the header is a JSON object where each JSON key is an environment variable and each JSON value is its corresponding value. If you specify system paths, use forward slashes (/).

```
com.ibm.websphere.jmx.connector.rest.transferCredentials=list_of_environment_variables
```

- Specify environment variables to set before the actions run. The payload of the header is a JSON object where each JSON key is an environment variable and each JSON value is its corresponding value. If you specify system paths, use forward slashes (/).

```
com.ibm.websphere.jmx.connector.rest.transferEnvVars=list_of_environment_variables
```

### 3. Delete a file from multiple registered hosts using the **DELETE** operation.

- a. Ensure the target location is within a configurable write directory of the hosts.
- b. Invoke the **DELETE** operation.

```
DELETE https://controller_host:controller_port/IBMJMXConnectorREST/file/{filePath}[?recursiveDelete=boolean]
```

- {*filePath*} is a UTF-8 URL-encoded absolute path that specifies the target location. For example, if the file is C:/temp/myFile.txt, the path is C%3A%2Ftemp%2FmyFile.txt.
- recursiveDelete is an optional query parameter that deletes non-empty directories. The default value is false.

## What to do next

Get status or details on the REST call. See “Getting status on a REST call for multiple registered hosts.”

## Getting status on a REST call for multiple registered hosts

8.5.5.6

You can get status or details about a call to upload or delete files on multiple registered hosts of a collective by invoking the **GET REST API**.

### Before you begin

Call a REST API to upload or delete files on multiple registered hosts. See “Uploading files to multiple registered hosts with a single REST call” on page 1037 or “Deleting files from multiple registered hosts with a single REST call” on page 1039.

### Procedure

- Get overall multiple task status, with the option of filtering the results.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/[?key=value]
```

The *key* variable is a task property filtered with the *value* variable. You can specify multiple *key=value* pairs to use as query filters.

- Get status about a specific task.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/{taskId}
```

{*taskId*} represents a task; for example, c8a2b96b-5a6a-493c-ad0c-140df87d61cf.

- Get a list of available properties from a specific task.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/{taskId}/properties
```

{*taskId*} represents a task; for example, c8a2b96b-5a6a-493c-ad0c-140df87d61cf.

- Get the value of a property in a specific task.



```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/{taskID}/properties/{property}
- {taskID} represents a task; for example, c8a2b96b-5a6a-493c-ad0c-140df87d61cf.
- {property} represents the property to get.
```

- Get status about the hosts of a specific task.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/{taskID}/hosts
{taskID} represents a task; for example, c8a2b96b-5a6a-493c-ad0c-140df87d61cf.
```

- Get details about steps that were taken within a specific host of a specific task.

```
GET https://controller_host:controller_port/IBMJMXConnectorREST/file/status/{taskID}/hosts/{hostName}
- {taskID} represents a task; for example, c8a2b96b-5a6a-493c-ad0c-140df87d61cf.
- {hostName} represents a host; for example, machineA.xyz.com.
```

## Example

Get overall multiple task status, with the option of filtering the results. If the JSON response is:

```
[
 {
 "taskID" : String ,
 "taskStatus" : String,
 "taskURL" : URL
 }*
]
```

For the following **GET** call:

```
GET https://myTarget.com:9443/IBMJMXConnectorREST/file/status
```

Typical status is:

```
[
 {
 "taskID" : "c8a2b96b-5a6a-493c-ad0c-140df87d61cf" ,
 "taskStatus" : "succeeded",
 "taskURL" : "/IBMJMXConnectorREST/file/status/c8a2b96b-5a6a-493c-ad0c-140df87d61cf"
 },
 {
 "taskID" : "18c807ff-e7bb-4584-a988-278039d0aabd" ,
 "taskStatus" : "failed",
 "taskURL" : "/IBMJMXConnectorREST/file/status/18c807ff-e7bb-4584-a988-278039d0aabd"
 }
]
```

For the following **GET** call with two *key=value* pairs:

```
GET https://myTarget.com:9443/IBMJMXConnectorREST/file/status?user=bob&status=succeeded
```

Typical status is:

```
[
 {
 "taskID" : "c8a2b96b-5a6a-493c-ad0c-140df87d61cf" ,
 "taskStatus" : "succeeded",
 "taskURL" : "/IBMJMXConnectorREST/file/status/c8a2b96b-5a6a-493c-ad0c-140df87d61cf"
 }
]
```

## Configuring binary logging in Liberty

Use this information as a guide for configuring binary logging in your Liberty.

### About this task

Binary logging provides faster log and trace handling capabilities and more flexible ways to use log and trace content than the default Liberty log and trace framework.

A server configuration consists of a `bootstrap.properties` file, a `server.xml` file, and any (optional) files that are included with those files. The `bootstrap.properties` file specifies properties that need to be available before the main configuration is processed, and are kept to a minimum. The `server.xml` file is the primary configuration file for the server.

The `server.xml` file and its associated files use a simple xml format that is suitable for most text editors.

**Distributed operating systems** A richer editing experience is provided by the eclipse server adapter for Liberty (WAS4D+ adapter), which uses a generated schema to provide drop-down lists of available choices, auto-completion, and other editing tools. For a description of the eclipse server adapter for Liberty, see “Editing the Liberty configuration by using developer tools” on page 938.

The `bootstrap.properties` file specifies whether the server uses binary logging as the log and trace framework, or the default log and trace framework. A server restart is required to switch between binary logging and the default log and trace framework.

You can modify the configuration of binary logging through the server configuration or the `bootstrap.properties` file.

- Server configuration: To get logging from your own code, which is loaded after server configuration processing, use the server configuration to configure binary logging.
- `bootstrap.properties` file: You might need to set logging properties to take effect before the server configuration files are processed. For example, if you need to analyze problems that occur early in server start or configuration processing. In this case, you can configure binary logging in the `bootstrap.properties` file.

You can set Logging properties in either the `bootstrap.properties` or the `server.xml` file. Use attributes in the `server.xml` file, or use equivalent properties in the `bootstrap.properties` file. Any settings in the `bootstrap.properties` file are used from the time the server reads the `bootstrap.properties` file until the time the `server.xml` file is processed. If the logging properties in the `bootstrap.properties` file are not replaced or reset in the `server.xml` file, the property values in the `bootstrap.properties` file continue to be used.

When binary logging is enabled, the `maxFileSize`, `maxFiles`, `messageFileName`, `traceFileName`, and `traceFormat` logging element attributes are ignored (since binary logging runs without `trace.log` and `messages.log` files). The `traceSpecification`, `consoleLogLevel`, and `logDirectory` attributes continue to be used to set the trace specification, the level for the console log, and the placement of the log and trace files.

If you set logging or binary logging attributes in the `server.xml` file, you can avoid changes in configuration between startup time and runtime by setting the corresponding properties in the `bootstrap.properties` file to the same value. If no logging or binary logging properties are set in the `bootstrap.properties` file, the server uses the default logging settings.

## Procedure

- Enable binary logging for the server by updating the `bootstrap.properties` file. In the `bootstrap.properties` file, add the following text on a line by itself:  
`websphere.log.provider=binaryLogging-1.0`
- Use the following parameters to configure binary logging. All subelements that are listed are subelements of the logging element in the `server.xml` file. The following table lists the attributes that are configurable in the `server.xml` file and the equivalent properties that can be set in the `bootstrap.properties` file:

Table 79. Binary logging attributes that are configurable in `server.xml` and the equivalent properties that can be set in `bootstrap.properties`

Logging subelement	Attribute	Equivalent <code>bootstrap.properties</code> property
binaryLog	purgeMaxSize	<code>com.ibm.hpel.log.purgeMaxSize</code>
	purgeMinTime	<code>com.ibm.hpel.log.purgeMinTime</code>
	fileSwitchTime	<code>com.ibm.hpel.log.fileSwitchTime</code>
	bufferingEnabled	<code>com.ibm.hpel.log.bufferingEnabled</code>
	outOfSpaceAction	<code>com.ibm.hpel.log.outOfSpaceAction</code>
binaryTrace	purgeMaxSize	<code>com.ibm.hpel.trace.purgeMaxSize</code>
	purgeMinTime	<code>com.ibm.hpel.trace.purgeMinTime</code>
	fileSwitchTime	<code>com.ibm.hpel.trace.fileSwitchTime</code>
	bufferingEnabled	<code>com.ibm.hpel.trace.bufferingEnabled</code>
	outOfSpaceAction	<code>com.ibm.hpel.trace.outOfSpaceAction</code>

The following example shows a `bootstrap.properties` file that is configured to enable binary logging:  
`websphere.log.provider=binaryLogging-1.0`

The following example shows a `server.xml` file with the binary logging subelements. The log content is set to expire after 96 hours and the trace content is set to retain a maximum of 1024MB:

```
<server description="new server">
 <logging>
 <binaryLog purgeMinTime="96"/>
 <binaryTrace purgeMaxSize="1024"/>
 </logging>
</server>
```

For the full logging configuration reference, see the `logging`, `binaryLog`, and `binaryTrace` elements in the `**** MISSING FILE ****`.

## Results

After you restart the server, binary logging is enabled and configured.

## Administering the transaction service on Liberty

Liberty provides many of the same transaction services that are provided in the WebSphere Application Server traditional. If an application uses two or more resources, the Liberty transaction manager coordinates the updates to all the resource managers under the control of a global transaction. In Liberty, transaction services are implicitly activated when you specify features that require the use of transactions; for example, `jpa-2.0`, `jdbc-4.0`, and `wasJmsServer-1.0`.

### About this task

You can control when database transaction recovery occurs, you can specify configuration settings that determine how database transactions are recovered, and you can choose whether to store your transaction logs as operating system files or in a relational database.

### Liberty: Configuring the startup of the transaction service

Database transaction recovery can occur either when the transaction service is first used, or at server startup.

## About this task

By default, transaction recovery after a server failure happens when the transaction service is first used rather than at server startup. You can alter this behavior by specifying transaction service attributes that control when recovery happens, and whether the system waits for recovery to finish before allowing transactional work to proceed.

## Procedure

To configure transaction service startup, specify the following attributes in the transaction element in the `server.xml` file:

- `recoverOnStartup`

This attribute can take the following values:

- `true`: Transaction recovery occurs at server startup.
- `false`: Transaction recovery occurs when the transaction service is first used.

- `waitForRecovery`

This attribute can take the following values:

- `true`: The server waits for transaction recovery to finish before allowing transactional work to proceed.
- `false`: The server allows transactional work to proceed without waiting for transaction recovery to finish.

## Example

With the following transaction element configuration, transaction recovery occurs at server startup, and the server waits for transaction recovery to finish before allowing transactional work to proceed.

```
<transaction
 recoverOnStartup="true"
 waitForRecovery="true"
/>
```

## How database transactions are recovered

When the Liberty transaction manager recovers indoubt database transactions, it uses either the unique identifier or the JNDI name to locate the current `dataSource` element, and then determines the user ID and password to use for recovery.

Configure a data source by specifying the attributes of the `dataSource` element in the `server.xml` configuration file. You can assign a unique identifier or a **jndiName** attribute for the data source as follows:

```
<dataSource id="ds1" jndiName="jdbc/ds1"... />
```

You must not change the value of the **id** or **jndiName** attribute when a recovery is pending for a transaction in which the data source participated. If you change any other attributes of the `dataSource` element, those changes are retained for the recovery. Therefore, you can, for example, add a **recoveryAuthDataRef** attribute that specifies a database user ID and password to use for recovery.

The database user ID and password to use for recovery are determined according to the following order of precedence:

1. If the `dataSource` element has the **recoveryAuthDataRef** attribute defined, then the user ID and password from the `authData` element are used. For example:

```
<authData id="recoveryAuth" user="dbuser1" password="{xor}0z0vKDtu"/>
<dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2"
 recoveryAuthDataRef="recoveryAuth" .../>
```

2. If container-managed authentication is used, then the user ID and password from the container-managed authentication alias are used. For example:
  - In the `ibm-web-bnd.xml` file, you have the following code:
 

```
<resource-ref name="jdbc/ds1ref" binding-name="jdbc/ds1">
 <authentication-alias name="user1Auth"/>
</resource-ref>
```
  - In the `server.xml` file, you must define the following code:
 

```
<authData id="user1Auth" user="dbuser1" password="{xor}0z0vKDtu"/>
<dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" .../>
```
3. The user ID and password from the `dataSource` element are used. For example:
 

```
<dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" ...>
 <properties.db2.jcc databaseName="testdb" user="dbuser1" password="{xor}0z0vKDtu"/>
</dataSource>
```
4. If none of the previous conditions are satisfied, and the recovery is attempted without any user ID and password, then the behavior is determined by the JDBC driver and database.

**Note:** If the transaction recovery is performed by an application-defined data source, such as an `@DataSourceDefinition` annotation or a `<data-source>` element in the deployment descriptor, you must ensure that the associated application is running when the recovery is taking place. You cannot use configuration settings in the `server.xml` file to recover application-defined data sources.

## Storing transaction logs in a relational database

You can choose to store your Liberty transaction logs in a relational database rather than as operating system files. In WebSphere Application Server traditional, this feature provides high availability (HA) support without having to use a shared file system. The feature is provided in Liberty for compatibility and for evaluation and testing purposes.

### About this task

The WebSphere Application Server transaction service writes information to a transaction log for every global transaction that involves two or more resources, or that is distributed across multiple servers. These transactions are started or stopped either by applications or by the container in which they are deployed. The transaction service maintains transaction logs to ensure the integrity of transactions. Information is written to the transaction logs in the prepare phase of a distributed transaction, so that if a WebSphere Application Server with active transactions restarts after a failure, the transaction service is able to use the logs to replay any indoubt transactions. This allows the overall system to be brought back to a consistent state.

In previous releases of WebSphere Application Server, the transaction logs were stored as operating system files. In WebSphere Application Server Version 8.5.5 and later, this remains the default configuration but you can choose to store the transaction logs in a relational database management system (RDBMS). This configuration option is aimed at customers working in an HA environment. In previous releases of WebSphere Application Server, HA transaction support required the use of a shared file system to host the transaction logs, such as an NFSv4-mounted network attached storage (NAS) or a storage area network (SAN). This new feature allows customers, particularly those with an investment in HA database technology, to use their HA database as a shared repository for the transaction logs, as an alternative to using a shared file system.

By default, Liberty transaction logs are stored in operating system files. However, for compatibility with WebSphere Application Server traditional, and for evaluation and testing purposes, you can configure the transaction logs to be stored in an RDBMS. You can use any database type that is supported by Liberty.

## Procedure

To configure the Liberty transaction logs to be stored in an RDBMS, complete the following steps:

1. Configure a dedicated, non-transactional data source in the Liberty `server.xml` file.

The following example extract from the `server.xml` file shows how to configure Liberty to store its transaction logs in a DB2 database:

```
<transaction>
 <dataSource transactional="false">
 <jdbcDriver libraryRef="DB2JCC4LIB"/>
 <properties.db2.jcc currentSchema="CBIVP"
 databaseName="SAMPLE" driverType="4"
 portNumber="50000" serverName="localhost"
 user="db2admin" password="{xor}0z1tPjsyNjE=" />
 </dataSource>
</transaction>

<library id="DB2JCC4LIB">
 <fileset dir="C:/SQLLIB/java" includes="db2jcc4.jar db2jcc_license_cu.jar"/>
</library>
```

2. (optional) Create the database tables.

Liberty attempts to create the necessary database tables when the server first starts. When this is not possible, due to insufficient permission for example, the server fails to start. Under these circumstances, you must create the two required database tables manually.

The following DDL structures show how to create the tables on DB2:

```
CREATE TABLE WAS_TRAN_LOG(
 SERVER_NAME VARCHAR(128),
 SERVICE_ID SMALLINT,
 RU_ID BIGINT,
 RUSECTION_ID BIGINT,
 RUSECTION_DATA_INDEX SMALLINT,
 DATA LONG VARCHAR FOR BIT DATA)

CREATE TABLE WAS_PARTNER_LOG(
 SERVER_NAME VARCHAR(128),
 SERVICE_ID SMALLINT,
 RU_ID BIGINT,
 RUSECTION_ID BIGINT,
 RUSECTION_DATA_INDEX SMALLINT,
 DATA LONG VARCHAR FOR BIT DATA)
```

The following DDL structures shows how to create the database table on Oracle:

```
CREATE TABLE WAS_TRAN_LOG(
 SERVER_NAME VARCHAR(128),
 SERVICE_ID SMALLINT,
 RU_ID NUMBER(19),
 RUSECTION_ID NUMBER(19),
 RUSECTION_DATA_INDEX SMALLINT,
 DATA BLOB)

CREATE TABLE WAS_PARTNER_LOG(
 SERVER_NAME VARCHAR(128),
 SERVICE_ID SMALLINT,
 RU_ID NUMBER(19),
 RUSECTION_ID NUMBER(19),
 RUSECTION_DATA_INDEX SMALLINT,
 DATA BLOB)
```

## Administering data access resources on Liberty

Administer data access resources using the Liberty features for JDBC, MongoDB, and more.

## Administering data access applications on Liberty

With Liberty, applications can access data in both relational and NoSQL databases.

### Procedure

- Configuring relational database connectivity in Liberty.
- Configure connection pooling for database connections
- Configure database transaction recovery

### Configuring relational database connectivity in Liberty:

You can configure a data source associated with different JDBC providers for database connectivity. The JDBC providers supply the driver implementation classes that are required for JDBC connectivity with your specific vendor database.

### About this task

To access a database from your application, you must use a data source. Data sources are provided by JDBC drivers and come in the following varieties:

- `javax.sql.DataSource`  
This is the basic form of a data source. It does not provide interoperability that enhances connection pooling, and cannot participate as a two-phase capable resource in transactions involving multiple resources.
- `javax.sql.ConnectionPoolDataSource`  
This type of data source is enabled for connection pooling. It cannot participate as a two-phase capable resource in transactions involving multiple resources.
- `javax.sql.XADataSource`  
This type of data source is both enabled for connection pooling and is able to participate as a two-phase capable resource in transactions involving multiple resources.

In order to be usable in the Liberty, your JDBC driver must provide at least one of these types of data sources. For the commonly used JDBC drivers, Liberty is already aware of the implementation class names for the various data source types. You only need to tell Liberty where to find the JDBC driver.

### Procedure

1. In the `server.xml` file, define a shared library pointing to the location of your JDBC driver JAR or compressed files. For example:

```
<library id="DB2JCC4Lib">
 <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

2. Define a data source using the JDBC driver. If you don't specify the type of data source, Liberty chooses the data source in the following order depending on which is available.

- `javax.sql.ConnectionPoolDataSource`
- `javax.sql.DataSource`
- `javax.sql.XADataSource`

Here is an example that accepts the default for data source type:

```
<dataSource id="db2" jndiName="jdbc/db2">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
</dataSource>
```

Here is an example that uses `javax.sql.XADataSource` type:

```
<dataSource id="db2xa" jndiName="jdbc/db2xa" type="javax.sql.XADataSource">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
</dataSource>
```

A default data source is available when at least one Java EE 7 feature is enabled. This data source uses a different priority to determine the type if none is specified.

- javax.sql.XADataSource
- javax.sql.ConnectionPoolDataSource
- javax.sql.DataSource

This data source is available as java:comp/DefaultDataSource. A jndiName does not need to be specified for it. To configure the default data source, specify a data source with the id set to DefaultDataSource. Here is an example that configures the default data source to point at a DB2 database:

```
<dataSource id="DefaultDataSource">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
</dataSource>
```

3. Optional: Configure attributes for the data source, such as JDBC vendor properties and connection pooling properties.

For example:

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2" connectionSharing="MatchCurrentState"
 isolationLevel="TRANSACTION_READ_COMMITTED" statementCacheSize="20">
 <connectionManager maxPoolSize="20" minPoolSize="5"
 connectionTimeout="10s" agedTimeout="30m"/>
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"
 currentLockTimeout="30s" user="user1" password="pwd1"/>
</dataSource>
```

For a full list of configuration attributes for the dataSource element, connectionManager element and some commonly used JDBC vendors, see “Data Source (dataSource)” on page 54.

4. Optional: Configure data sources for commonly used databases according to the following examples.

#### For DB2

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
</dataSource>

<library id="DB2JCC4Lib">
 <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

#### For DB2 on iSeries (Native)

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2iNative">
 <jdbcDriver libraryRef="DB2iNativeLib"/>
 <properties.db2.i.native databaseName="*LOCAL"/>
</dataSource>

<library id="DB2iNativeLib">
 <fileset dir="/QIBM/Proddata/java400/jdk6/lib/ext" includes="db2_classes16.jar"/>
</library>
```

#### For DB2 on iSeries (Toolbox)

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2iToolbox">
 <jdbcDriver libraryRef="DB2iToolboxLib"/>
 <properties.db2.i.toolbox databaseName="SAMPLEDB" serverName="localhost"/>
</dataSource>

<library id="DB2iToolboxLib">
 <fileset dir="/QIBM/ProdData/Http/Public/jt400/lib" includes="jt400.jar"/>
</library>
```



### For Derby Embedded

```
<dataSource id="DefaultDataSource" jndiName="jdbc/derbyEmbedded">
 <jdbcDriver libraryRef="DerbyLib"/>
 <properties.derby.embedded databaseName="C:/databases/SAMPLEDB" createDatabase="create"/>
</dataSource>

<library id="DerbyLib">
 <fileset dir="C:/db-derby-10.8.1.2-bin/lib"/>
</library>
```

### For Derby Network Client

```
<dataSource id="DefaultDataSource" jndiName="jdbc/derbyClient">
 <jdbcDriver libraryRef="DerbyLib"/>
 <properties.derby.client databaseName="C:/databases/SAMPLEDB" createDatabase="create"
 serverName="localhost" portNumber="1527"/>
</dataSource>

<library id="DerbyLib">
 <fileset dir="C:/db-derby-10.8.1.2-bin/lib"/>
</library>
```

### For Informix® JCC

```
<dataSource id="DefaultDataSource" jndiName="jdbc/informixjcc">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.informix.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="1526"/>
</dataSource>

<library id="DB2JCC4Lib">
 <fileset dir="C:/Drivers/jcc/4.8" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

### For Informix JDBC

```
<dataSource id="DefaultDataSource" jndiName="jdbc/informix">
 <jdbcDriver libraryRef="InformixLib"/>
 <properties.informix databaseName="SAMPLEDB" ifxIFXHOST="localhost"
 serverName="ol_machinename" portNumber="1526"/>
</dataSource>

<library id="InformixLib">
 <fileset dir="C:/Drivers/informix" includes="ifxjdbc.jar ifxjdbcx.jar"/>
</library>
```

### For Microsoft SQL Server (Microsoft JDBC driver)

```
<dataSource id="DefaultDataSource" jndiName="jdbc/mssqlserver">
 <jdbcDriver libraryRef="MSJDBC4Lib"/>
 <properties.microsoft.sqlserver databaseName="SAMPLEDB"
 serverName="localhost" portNumber="1433"/>
</dataSource>

<library id="MSJDBC4Lib">
 <fileset dir="C:/sqljdbc_6.0/enu/sqljdbc41.jar"/>
</library>
```

### For Microsoft SQL Server (DataDirect Connect for JDBC driver)

```
<dataSource id="DefaultDataSource" jndiName="jdbc/ddsqlserver">
 <jdbcDriver libraryRef="DataDirectLib"/>
 <properties.datadirect.sqlserver databaseName="SAMPLEDB"
 serverName="localhost" portNumber="1433"/>
</dataSource>

<library id="DataDirectLib">
 <fileset dir="C:/DataDirect/Connect-4.2/lib/sqlserver.jar"/>
</library>
```

### For MySQL

```
<dataSource id="DefaultDataSource" jndiName="jdbc/mysql">
 <jdbcDriver libraryRef="MySQLLib"/>
 <properties databaseName="SAMPLEDB" serverName="localhost" portNumber="3306"/>
</dataSource>
```

```
<library id="MySQLLib">
 <fileset dir="C:/mysql-connector-java-x.x.xx/mysql-connector-java-x.x.xx.jar"/>
</library>
```

#### For Oracle

```
<dataSource id="DefaultDataSource" jndiName="jdbc/oracle">
 <jdbcDriver libraryRef="OracleLib"/>
 <properties.oracle URL="jdbc:oracle:thin:@//localhost:1521/SAMPLEDB"/>
</dataSource>

<library id="OracleLib">
 <fileset dir="C:/Oracle/lib/ojdbc6.jar"/>
</library>
```

#### For Sybase

```
<dataSource id="DefaultDataSource" jndiName="jdbc/sybase">
 <jdbcDriver libraryRef="SybaseLib"/>
 <properties.sybase databaseName="SAMPLEDB" serverName="localhost" portNumber="5000"/>
</dataSource>

<library id="SybaseLib">
 <fileset dir="C:/Drivers/sybase/jconn4.jar"/>
</library>
```

#### For solidDB

```
<dataSource id="DefaultDataSource" jndiName="jdbc/solidDB">
 <jdbcDriver libraryRef="solidLib"/>
 <properties databaseName="SAMPLEDB" URL="jdbc:solid://localhost:2315"/>
</dataSource>

<library id="solidLib">
 <fileset dir="C:/Drivers/solidDB/SolidDriver2.0.jar"/>
</library>
```

#### For a JDBC driver that is not known to Liberty

```
<dataSource id="DefaultDataSource" jndiName="jdbc/sample" type="javax.sql.XADataSource">
 <jdbcDriver libraryRef="SampleJDBCLib"
 javax.sql.XADataSource="com.ibm.sample.SampleXADataSource"/>
 <properties databaseName="SAMPLEDB" hostName="localhost" port="12345"/>
</dataSource>

<library id="SampleJDBCLib">
 <fileset dir="C:/Drivers/SampleJDBC/sampleDriver.jar"/>
</library>
```

In the example, the JDBC driver is located at `C:/Drivers/SampleJDBC/sampleDriver.jar` and provides an implementation of `javax.sql.XADataSource` named `com.ibm.sample.SampleXADataSource`. The JDBC driver also provides vendor-specific data source properties such as **databaseName**, **hostName** and **port**.

Configuring a default data source: 8.5.5.6

You can configure a default data source that is associated with different JDBC providers for database connectivity. The JDBC providers supply the driver implementation classes that are required for JDBC connectivity with your specific vendor database.

#### About this task

To access a database from your application, you must configure a data source.

#### Procedure

1. Configure the `datasource` element with the ID `DefaultDataSource` in the `server.xml` file.

```

<dataSource id="DefaultDataSource">
 <jdbcDriver libraryRef="MyJDBCLib"/>
 <properties.derby.embedded databaseName="myDB" createDatabase="create"/>
 <containerAuthData user="user1" password="{xor}0z0vKDTu" />
</dataSource>
<library id="MyJDBCLib">
 <file name="C:/derby/derby.jar"/>
</library>

```

**Note:** The server must be running at the Java Enterprise Edition 7 platform level. This platform level is enabled when one or more Java Enterprise Edition 7 features are enabled in the `server.xml` file.

- To use the `DefaultDataSource` in a web application, a reference can be obtained with dependency injection:

```

@Resource
DataSource defaultDataSource;

```

or through JNDI lookup:

```

DataSource defaultDataSource = (DataSource) new InitialContext().lookup("java:comp/DefaultDataSource");

```

*How data source configuration updates are applied:*

If you change the attributes of the **dataSource** element while a server is running, the updates to different attributes are applied at different times and in different ways.

You configure a data source by specifying the attributes of the **dataSource** element in the `server.xml` configuration file. If you change these attributes for a running server, the updates are applied at different times and in different ways, depending on which attribute is changed. The following table describes, for each attribute of the **dataSource** element, how a configuration change is applied at run time.

*Table 80. How data source configuration updates are applied at run time.* The first column of the table lists the attributes of the **dataSource** element. The second column describes, for each attribute, how the configuration update is applied at run time.

Attribute name	How the configuration update is applied
<b>beginTranForResultSetScrollingAPIs</b>	The update is effective immediately.
<b>beginTranForVendorAPIs</b>	The update is effective immediately.
<b>commitOrRollbackOnCleanup</b>	The update is effective immediately.
<b>connectionManagerRef</b>	All connections and the connection pool are destroyed. The data source is then managed by the new connection manager.
<b>connectionSharing</b>	The update is applied with each first connection handle in a transaction.
<b>isolationLevel</b>	The update is applied with new connection requests; current connections retain their isolation level.
<b>jdbcDriverRef</b>	All connections and the connection pool are destroyed. The new JDBC driver is then used.
<b>jndiName</b>	All connections and the connection pool are destroyed. The new JNDI name is then used.
<b>propertiesRef</b>	If the data source is Derby Embedded, all connections and the connection pool are destroyed before new properties go into effect. For other JDBC drivers, the new properties go into effect with new connection requests.
<b>queryTimeout</b>	The update is effective immediately.
<b>recoveryAuthDataRef</b>	Authentication data for transaction recovery. All connections and the connection pool are destroyed. The new recovery authentication data is then used.
<b>statementCacheSize</b>	The statement cache is resized upon next use.

Table 80. How data source configuration updates are applied at run time (continued). The first column of the table lists the attributes of the `dataSource` element. The second column describes, for each attribute, how the configuration update is applied at run time.

Attribute name	How the configuration update is applied
<code>supplementalJDBCTrace</code>	All connections and the connection pool are destroyed. The new setting is then used.
<code>syncQueryTimeoutWithTransactionTimeout</code>	The update is effective immediately.
<code>transactional</code>	The update is applied to new connections and existing connections not in use from the connection pool.
<code>type</code>	All connections and the connection pool are destroyed. The new setting is then used.

*Application-defined data sources:*

You can define a data source within your application, through annotations or in the deployment descriptor, as defined by the Java EE specification.

**Note:** The `commonLibraryRef` class loader attribute is recommended for application defined data sources. The `privateLibraryRef` attribute cannot be used for the `java:global` namespace and is discouraged for the other scopes. If multiple applications declare the same `java:global` namespace to specify the data source, the `server.xml` files of the applications must all specify a `commonLibraryRef` attribute to the same shared library.

When defining a data source in an application, the JDBC driver must be made available to the application. This is accomplished by configuring a shared library in the `server.xml` for your application.

For example:

```
<application id="myApp" name="myApp" location="myApp.war" type="war">
 <classloader commonLibraryRef="DB2Lib"/>
</application>

<library id="DB2Lib">
 <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

Then, you can define a data source in your application either through annotations or in the deployment descriptor.

- Use annotations as in the following example:

```
@DataSourceDefinition(
 name = "java:comp/env/jdbc/db2",
 className = "com.ibm.db2.jcc.DB2DataSource",
 databaseName = "SAMPLEDB",
 serverName = "localhost",
 portNumber = 50000,
 properties = { "driverType=4" },
 user = "user1",
 password = "pwd1"
)
```

```
public class MyServlet extends HttpServlet {

 @Resource(lookup="java:comp/env/jdbc/db2")
 DataSource ds;
```

- Use the deployment descriptor as in the following example, for example, in a `web.xml` file:

```

<data-source>
 <name>java:comp/env/jdbc/db2</name>
 <class-name>com.ibm.db2.jcc.DB2DataSource</class-name>
 <server-name>localhost</server-name>
 <port-number>50000</port-number>
 <database-name>SAMPLEDB</database-name>
 <user>user1</user>
 <password>pwd1</password>
 <property><name>driverType</name><value>4</value></property>
</data-source>

```

In general, properties that can be defined on `dataSource` or `connectionManager` in the `server.xml` files can also be specified on application defined data sources. Two exceptions to this are `connectionManagerRef` and `jdbcDriverRef`, which you cannot specify because the application defined data source implicitly defines the connection manager and JDBC driver. When using application defined data sources for two-phase commit, you can specify the `recoveryAuthDataRef` property to select the authentication data that is used for transaction recovery. However, it is important to be aware that recovery of transactions is only possible while the application is running. You can use variables, encoded passwords, and duration syntax in application defined data sources.

**Note:** The duration syntax does not apply to properties that are explicitly defined in the annotation, such as `loginTimeout` or `maxIdleTime`.

Here is an example of two data sources using connection manager properties, variables, encoded passwords, and duration syntax.

```

@DataSourceDefinitions(value = {
 @DataSourceDefinition(
 name = "java:comp/env/jdbc/derby",
 className = "org.apache.derby.jdbc.EmbeddedDataSource40",
 databaseName = "${shared.resource.dir}/data/SAMPLEDB",
 minPoolSize = 1,
 maxPoolSize = 10,
 maxIdleTime = 180,
 properties = { "agedTimeout=10m", "connectionTimeout=30s", "createDatabase=create" }
),
 @DataSourceDefinition(
 name = "java:comp/env/jdbc/oracle",
 className = "oracle.jdbc.pool.OracleDataSource",
 url = "jdbc:oracle:thin:@//localhost:1521/SAMPLEDB",
 user = "user1",
 password = "{xor}0z0vKDtt"
)
})

```

*Configuring client reroute for applications that use DB2 databases:* 8.5.5.7

You can use the client reroute feature to configure your enterprise applications for a DB2 database to recover from a communication loss, and the applications can continue to work with minimal interruption. Rerouting is central to the support of continuous operations, but rerouting is only possible when there is an alternative location that is identified to the application server connection.

### Before you begin

This task assumes that:

- The DB2 data source to which your application connects is running:
  - DB2 for z/OS Version 10.1 or later or
  - DB2 Database for Linux, UNIX, and Windows Version 9.7 or later

- You configured the DB2 database with a redundant setup or the ability to fail the DB2 server to a standby node.

### About this task

You can use client reroute for DB2 to provide information about alternative servers in case the connection to the primary database server fails.

Without any configuration on the client side, the Java Common Connectivity (JCC) Java Database Connectivity (JDBC) driver for DB2 supports the client reroute capability, if it is enabled on the DB2 server, when the driver makes an initial connection to the DB2 server. When the JCC JDBC driver connects to a DB2 server that has one or more alternative servers that are configured, the primary server sends information about the alternative servers to the JCC JDBC driver. If the connection to the primary server fails, the JCC JDBC driver is able to reroute connections to an alternative server. If the application server process crashes, however, the alternative server information is lost and the client needs to connect to the primary server again. If the client cannot make an initial connection to the primary server, the client has no knowledge of the alternative servers and cannot reroute.

To overcome this problem, you can configure a DB2 data source in the application server with the **Alternate server name** and **Alternate port number** fields, or with the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` data source custom properties, to support client reroute even on the initial connection attempt. If the JDBC driver is not able to connect to the primary DB2 server, the information that is necessary for a client reroute is already present, and the JDBC driver can reroute the connection to an alternate server.

**Attention:** The data source custom property, `enableClientAffinitiesList`, changes the semantics of the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` properties.

When a connection is rerouted and the JCC JDBC driver is connected to the alternative DB2 server, the alternative server sends information about its own alternative server to the JCC JDBC driver. The JCC JDBC driver then has the information that is required to reroute the connection again if the alternative DB2 server is not available. The server that was originally the alternative server is now the primary server, and a new alternative server is established. However, this new state of the primary and alternative servers is no longer kept by the JCC JDBC driver. If the application server fails and is restarted, the JCC JDBC driver must start from the original server configuration and attempt to connect to the server that was originally considered the primary server.

You can use the automatic client rerouting feature within the following DB2 configurable environments:

- Enterprise Server Edition (ESE) with the data partitioning feature (DPF)
- Data Propagator (DPROPR)-style replication
- High availability cluster multiprocessor (HACMP<sup>™™</sup>)
- High availability disaster recovery (HADR)

### Procedure

1. Define your DB2 data source in the `server.xml` file with the following properties:

- `clientRerouteAlternateServerName`
  - Type of valid values:
    - Domain name; for example: `www.ibm.com`
    - IP address (IPv4 and IPv6); for example: `23.72.11.219`
  - Format of multiple values:
    - Comma separated; for example: `host1, host2, host3`
    - Space separated; for example: `host1 host2 host3`
  - Order significance:

- The order of the provided host names is the order that the JCC JDBC driver uses to try to find the next available server to connect to.
  - `clientRerouteAlternatePortNumber`
    - Type of valid values:
      - Integer representing the port number; for example: 50000
    - Format of multiple values:
      - Comma separated; for example: port1, port2, port3
      - Space separated; for example: port1 port2 port3
    - Order significance:
      - The order of the provided ports must match the order of their associated servers.
2. (Optional) You can add one or both of the following properties:
- `retryIntervalForClientReroute`

This property defines the number of seconds the JCC JDBC driver waits between each attempt to establish a connection.

If no value is assigned, the default behavior is used. To learn more, see JDBC and SQL support.
  - `maxRetriesForClientReroute`

This property defines the number of retries the JCC JDBC driver attempts to connect the server before it decides to move on to the next server. This property is only used when `RetryIntervalForClientReroute` property is set.

If no value is assigned, the default behavior is used. To learn more, see JDBC and SQL support.

### Example

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2">
 <properties.db2.jcc
 databaseName="sampleDatabase"
 driverType="4"
 serverName="host"
 portNumber="50000"
 clientRerouteAlternateServerName="host01, host02, host03"
 clientRerouteAlternatePortNumber="50000, 50005, 50000"
 retryIntervalForClientReroute="2"
 maxRetriesForClientReroute="3" />
 ...
</dataSource>
```

**Note:** Ensure that an equal number of entries is specified for both ports and hosts. Otherwise, a warning is displayed and client reroute is not enabled.

*Configuring connection pooling for database connections:*

You can configure connection pooling for your data source by defining a connection manager for it.

### Example

The following example code uses the `connectionManager` element in the `server.xml` file to define a connection pool for a data source:

```
<dataSource id="DefaultDataSource" jndiName="jdbc/example" jdbcDriverRef="DB2" >
 <connectionManager maxPoolSize="10" minPoolSize="2"/>
 <properties.db2.jcc databaseName="TESTDB"/>
</dataSource>
```

The server uses default values for any connection management settings that are not defined on the connection manager element. If a connection manager is not defined at all for a data source, the server uses default values for all of the settings.

Using thread local storage for connections can increase performance for applications on multi-threaded systems. See Chapter 11, “Tuning Liberty,” on page 1445.

You can define multiple data sources and associate each with a different connection manager. However, you cannot associate multiple data sources with a single connection manager.

For more information about the `connectionManager` element, see \*\*\*\* MISSING FILE \*\*\*\*.

*How connection pooling configuration updates are applied:*

If you change the attributes of the `connectionManager` element while a server is running, the updates to different attributes are applied at different times and in different ways.

You configure a connection pool by specifying the attributes of the `connectionManager` element in the `server.xml` configuration file. If you change these attributes for a running server, the updates are applied at different times and in different ways, depending on which attribute is changed. The following table describes, for each attribute of the `connectionManager` element, how a configuration change is applied at run time.

*Table 81. How connection manager configuration updates are applied at run time.* The first column of the table lists the attributes of the `connectionManager` element. The second column describes, for each attribute, how the configuration update is applied at run time.

Attribute name	How the configuration update is applied
<code>agedTimeout</code>	The update is effective immediately.
<code>connectionTimeout</code>	The update is effective immediately.
<code>maxIdleTime</code>	The update is effective immediately.
<code>maxNumberOfMCsAllowableInThread</code>	The update is effective immediately.
<code>maxPoolSize</code>	The update is effective immediately.
<code>minPoolSize</code>	The update is effective immediately.
<code>numConnectionsPerThreadLocal</code>	The update is effective immediately.
<code>reapTime</code>	The update is effective immediately.
<code>purgePolicy</code>	The update is effective immediately.

**Note:** The attributes `agedTimeout` and `maxIdleTime` are updated immediately. However, they are not used fully unless the value of `reapTime` attribute is greater than zero.

Because updates to the connection manager are effective immediately, errors might occur if you make changes with active connections; including the potential risks for the connections to be ended prematurely.

#### Configuring CouchDB connectivity with the ektorp client library in Liberty: 8.5.5.4

Applications that run on Liberty can use CouchDB. For access to a CouchDB instance, applications can configure a connector for the NoSQL database with the ektorp client library.

#### Before you begin

Liberty provides configuration support for CouchDB. CouchDB is a scalable, high-performance, open source NoSQL database.

You must use Version 1.4.1 or later of the ektorp Java driver. Use the Maven plug-in to obtain the ektorp driver and its dependencies.



```

<dependency>
 <groupId>org.ektorp</groupId>
 <artifactId>org.ektorp</artifactId>
 <version>1.4.1</version>
</dependency>

```

## About this task

To enable an application to use CouchDB, you must configure a shared library for the CouchDB Java driver and a library reference to the shared library in the `server.xml` file. An application can access CouchDB either directly from the application, or through the `couchdb-1.0` feature and CouchDB instance configurations in the `server.xml` file.

## Procedure

1. Install the CouchDB Java driver in a location that your application and the Liberty runtime can access. For example, place the `ektorp` driver file and its dependencies in the `Liberty_profile_root/usr/servers/server_name/lib` directory.
2. Configure a shared library for the `ektorp` driver files in the `server.xml` file of the Liberty server.

```

<library id="couchdb-lib">
 <fileset
 dir='${server.config.dir}/lib'
 includes='org.ektorp-1.4.1.jar
 commons-codec-1.6.jar
 commons-io-2.0.1.jar
 commons-logging-1.1.1.jar
 httpClient-4.2.5.jar
 httpClient-cache-4.2.5.jar
 httpcore-4.2.4.jar
 jackson-annotations-2.2.2.jar
 jackson-core-2.2.2.jar
 jackson-databind-2.2.2.jar
 slf4j-api-1.6.4.jar
 slf4j-simple-1.6.4.jar' />
</library>

```

3. Enable your application to access CouchDB, either by direct access from the application or by using the `couchdb-1.0` feature.
  - Enable direct access to CouchDB from the application.
    - a. Configure a library reference for the shared library in an application element in the `server.xml` file.

```

<application ...>
 <classloader commonLibraryRef="couchdb-lib"/>
</application>

```

The application can now access the CouchDB APIs directly. If you want the application to use the runtime injection engine, continue with the next steps.

- Configure the `couchdb-1.0` feature, and the `couchdb` elements in the `server.xml` file.
  - a. Add the `couchdb-1.0` feature to the `server.xml` file.

```

<featureManager>
 <feature>couchdb-1.0</feature>
 <feature>jndi-1.0</feature>
</featureManager>

```

The JNDI feature is only required when you use JNDI to look up resources. This feature is not required if you use resource injection.

- b. Configure a `couchdb` element that has a reference to the shared library created in a previous step.

```
<couchdb id="couchdb" jndiName="couchdb/connector"
 libraryRef="couchdb-lib" url="http://example.com:5984" username="username"
 password="password"/>
```

Configuring a JNDI name enables an application or the Liberty runtime to look up the CouchDB instance.

- c. Enable your application to access CouchDB.

The following example shows both JNDI lookup and resource injection:

```
public class TestServlet extends HttpServlet {
 @Resource(name = "couchdb/connector")
 protected CouchDbInstance db;
 ...
 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException {
 // Alternatively use InitialContext lookup
 CouchDbInstance lookup = (CouchDbInstance) new
 InitialContext().lookup("java:comp/env/couchdb/connector");
 ...
 }
}
```

- d. If you are using JNDI lookup, add a resource environment reference to the web.xml file of your application:

```
<resource-env-ref>
 <resource-env-ref-name>couchdb/connector</resource-env-ref-name>
 <resource-env-ref-type>org.ektorp.CouchDbInstance</resource-env-ref-type>
</resource-env-ref>
```

You can use the couchdb-1.0 feature to configure a connection to an online Cloudant service. Specify the URL, userid, and password of your existing Cloudant account in the couchdb configuration element. For example:

```
<couchdb id='couchdb' jndiName='couchdb/connector' libraryRef='couchdb-lib' url='https://mylink.cloudant.com/' userna
```

See the documentation for “The limits to protection through password encryption” on page 616 to learn about how to secure passwords in configuration files.

## What to do next

Now that you have configured your application to enable the use of CouchDB, you are ready to test the use of CouchDB from your application.

## Administering web applications on Liberty

Liberty provides support to the web applications using Liberty features such as servlet-3.0, servlet-3.1, jsp-2.2, and other features.

### Procedure

Specify when servlets are loaded and initialized.

### Specifying when servlets are loaded and initialized

By default, Liberty defers servlet loading until a request is received for the associated web application. You can override this default behavior by specifying the web container **deferServletLoad** attribute to false.

### About this task

The servlet specification defines the **load-on-startup** servlet attribute, which is specified in the web.xml file of a web application. If a servlet has a non-negative value for the **load-on-startup** attribute, the servlet must be loaded and initialized when the web application is deployed. Liberty optimizes server start time and memory use by not starting a servlet until a request is received for the web application. You can override this deferral so that your servlets are loaded and initialized when the web application is

installed, rather than waiting for the first request for the application.

## Example

To configure the server to load servlets when a web application is installed, add the following line to the `server.xml` configuration file or a file that it includes:

```
<webContainer deferServletLoad="false"/>
```

This setting applies to all web applications installed in the server.

## Configuring Cross Origin Request Sharing on a Liberty server

8.5.5.9

You can enable Cross Origin Request Sharing (CORS) for your web applications on a Liberty server.

### About this task

Enabling CORS will allow JavaScript clients to make requests against your application on the Liberty server even if the client and the server are on two different domains. Web browsers prevent these requests due to same-origin policy.

### Procedure

1. Ensure the server configuration has all features needed for your deployed application, such as `servlet-3.0`, `jaxrs-1.1`, and so on. Also ensure the ports and user registry settings are correct for the deployed application.
2. Add the CORS service setting to the `server.xml` file. The `cors` element defines the CORS settings for the URL being setup in the domain.

### Example

Here is an example of a CORS configuration for a web application at the `sampleApp/path` context root.

```
<cors domain="/sampleApp/path"
 allowedOrigins="https://alice.com:8090"
 allowedMethods="GET, DELETE, POST"
 allowedHeaders="accept, MyRequestHeader1"
 exposeHeaders="MyResponseHeader1"
 allowCredentials="true"
 maxAge="3600" />
```

Here is an example of a CORS configuration for the RESTful endpoint `/ibm/api/collective`. This setting will apply to all the endpoint paths which start with `/ibm/api/collective` including `ibm/api/collective/docs`.

```
<cors domain="/ibm/api/collective"
 allowedOrigins="https://alice.com:8090"
 allowedMethods="GET, DELETE, POST"
 allowedHeaders="accept, MyRequestHeader1"
 exposeHeaders="MyResponseHeader1"
 allowCredentials="true"
 maxAge="3600" />
```

## Configuring Liberty for Servlet 3.1

You can configure Liberty for the Servlet 3.1 feature, which provides full support for the Servlet 3.1 specification.

## About this task

To configure a Liberty server to run an application that is enabled for Servlet 3.1, you must set the `<servlet-3.1>` feature.

## Procedure

Update the `server.xml` file to add the `<servlet-3.1>` feature. For example:

```
<featureManager>
 <feature>servlet-3.1</feature>
</featureManager>
```

### Important:

- The `websocket-1.0` and `websocket-1.1` features require the `servlet-3.1` feature and as a result, configuring the `websocket-1.0` or `websocket-1.1` feature causes the `servlet-3.1` feature to load.
- You can use Java EE 6 features, such as `jsp-2.2` and `jsf-2.0`, with the `servlet-3.1` feature. However, you cannot use a Java EE 6 feature to exploit Servlet 3.1 features.
- You can choose between the Servlet 3.0 and Servlet 3.1 feature implementations for each server instance, but you must consider any behavior changes. If required behavior is contained only in the Servlet 3.1 feature, then you must use the Servlet 3.1 feature. If an existing application would be adversely impacted by behavior changes in the Servlet 3.1 feature, then use the Servlet 3.0 feature to preserve the existing behavior for that application.
- It is not possible to use both the Servlet 3.0 and Servlet 3.1 features in the same Liberty server. If both features are configured, it will produce an error. Read the Servlet 3.1 behavior changes topic to learn about changes from Servlet 3.0 and Servlet 3.1.

## Results

The Servlet-3.1 feature is enabled and loads in the Liberty server at run time.

### Servlet 3.1 behavior changes:

The Servlet 3.1 implementation contains behavior changes that might cause an application that was written for Servlet 3.0 to behave differently or fail when you use the Servlet 3.1 feature.

You can choose between the Servlet 3.0 and Servlet 3.1 feature implementations for each server instance, with consideration for behavior changes. If the required behavior is contained in the Servlet 3.1 feature only, then you must use the Servlet 3.1 feature. If an existing application would be adversely impacted by behavior changes in the Servlet 3.1 feature, then using the Servlet 3.0 feature preserves the existing behavior for that application. It is not possible to use both the Servlet 3.0 and Servlet 3.1 features in the same server. It is an error to configure both features. If you configure both features, neither servlet feature is loaded.

The behavior changes are introduced for three reasons:

- Changes that are required by clarifications in the Servlet 3.1 specification.
- Changes that are required for the Servlet 3.1 implementation to pass the Servlet 3.1 Technology Compatibility Kit (TCK).
- Changes to improve the servlet implementation.

### Programmatically added servlets, filters, and listeners

A clarification from the Servlet 3.1 specification now makes it illegal for a `ServletContextListener` to programmatically configure servlets, filters, or listeners if the `ServletContextListener` was not declared in the `web.xml` file or `web-fragment.xml` file or was not annotated with `@WebListener`. As a result, any call

on the ServletContext to perform such programmatic configuration results in an UnsupportedOperationException.

### **Forward after asynchronous processing is started**

In the Servlet 3.0 implementation, a response is always closed before the forward method of the RequestDispatcher interface returns. However, due to a clarification in the Servlet 3.1 specification, the Servlet 3.1 implementation does not close or flush the response before the forward method of the RequestDispatcher interface returns, if the request is put into the asynchronous mode. This change might affect existing 3.0 applications, which add response output on return from forward because such response data is now sent, whereas in Servlet 3.0, it was not.

### **URL pattern clashes**

In Servlet 3.0, an application would start successfully even when a URL pattern was mapped to multiple servlets. However, due to a clarification in the Servlet 3.1 specification, the application must fail to start. In the Liberty Servlet 3.1 implementation, a message is output and the application fails to start:

```
SRVE9016E: Unable to insert mapping [{0}] for servlet named [{1}]. The URL pattern is already defined for servlet named
```

Explanation: There is an application error. A servlet mapping URL pattern should not map to multiple servlets.

User action: Change the URL pattern for the servlet mapping.

### **ServletContext.getMinorVersion()**

In the Servlet 3.0 feature implementation, this API returns 0.

In the Servlet 3.1 feature this API now returns 1.

### **ServletContext.getServerInfo()**

In the Servlet 3.0 feature implementation, this API returns SMF WebContainer.

In the Servlet 3.1 feature this API now returns IBM WebSphere Liberty/8.5.5.<x>, where <x> is the WebSphere Application Server fix pack number.

### **ServletResponse.reset()**

You can use ServletResponse.reset() to clear any buffered response data, the status code, and response headers when a response is not already committed. If the Servlet 3.1 feature is being used, this method also clears any record of ServletResponse.getWriter() or ServletResponse.getOutputStream() that were previously called.

### **X-Powered-By header**

In the Servlet 3.0 feature implementation, the X-Powered-By header is set to Servlet/3.0. In the Servlet 3.1 feature implementation, the X-Powered-By header is set to Servlet/3.1.

### **Resource reference injection target merging**

In the Servlet 3.0 specification, the <injection-target> elements of a resource reference that is defined in a web-fragment.xml file are added only to the parent web.xml file if the web.xml resource reference definition with the same name has no <injection-target> elements. In the Servlet 3.1 specification, it is clarified that all <injection-target> elements in web-fragment.xml descriptors are added to the parent web.xml descriptors list of <injection-target> elements for a resource reference of the same name. When the Servlet 3.1 feature is in use, this feature might change existing application function by activating

injection targets that were previously excluded from the web.xml file.

### **Tolerance of duplicate elements in web descriptors**

In the Servlet 3.1 specification, it was clarified that a web.xml file cannot contain two <absolute-ordering> elements. Deployment of an application with multiple <absolute-ordering> elements fails. Additionally, web-fragment.xml descriptors cannot contain two <ordering> elements. Deployment of an application with multiple <ordering> elements fails. Previously, the deployment would not fail, but the function of the elements might be indeterminate.

### **Web fragment ordering change in metadata - complete cases**

The processing of the <absolute-ordering> element is changed in cases where a web.xml descriptor is marked metadata-complete="true". Previously in metadata-complete="true" cases, all web fragment archives would be used. When the Servlet-3.1 feature is in use, the <absolute-ordering> element in metadata-complete cases is considered to be complete. This change results in fragments that are not listed in the <absolute-ordering> element to be excluded from processing.

### **AsyncContext.dispatch()**

When AsyncContext.dispatch() is used (for example, with no parameters), the request is dispatched to the original URL. With the Servlet-3.0 feature in use, if a query string was included with the original request, this is made available to the dispatched resource. However, when the Servlet 3.1 feature in use, if a query string was provided to the dispatching resource, it is this query string that is made available to the dispatched resource. For example:

Request for /FirstResource?param=One

First Resource:

```
getParameter("param") returns "One"
forward request to /SecondResource?param=Two
```

SecondResource

```
getParameter(param) returns "Two"
ac.start()
ac.dispatch() dispatches to /FirstResource
```

First Resource

```
Servlet-3.0 feature : getParamter("param") returns "One"
Servlet-3.1 feature : getParameter("param") returns "Two"
```

This change was required by the Servlet 3.1 TCK.

Obtaining the request or response object after an AsyncContext.dispatch() or AsyncContext.complete() is not permitted and results in the following exception being thrown:

```
java.lang.IllegalStateException: SRVE9015E: Cannot obtain the request or response object after an AsyncContext.dispatch() o
at com.ibm.ws.webcontainer31.async.AsyncContext31Impl.getRequest(AsyncContext31Impl.java:72)
[...]
```

### **SessionCookieConfig.setComment()**

According to the Java Servlet 3.1 Specification, this API returns an illegalStateException if it is called after the ServletContext completes initialization, and the Servlet 3.1 feature follows this required behavior. However, the Servlet 3.0 feature does not prevent use of this API after the context is initialized and as a result, applications that depend on the Servlet 3.0 feature behavior will not work with the Servlet 3.1 feature.

### **sendRedirect(java.lang.String location) API**

The sendRedirect(java.lang.String location) API accepts relative URLs; however, the servlet container must convert the relative URL to an absolute URL before it can send the response to the client. If the

location is relative without a leading '/' (folder/default.jsp), the container interprets it as relative to the current request URI. If the location is relative with a leading '/', the container interprets it as relative to the servlet container root.

For example, if the redirection location that is provided by application is folder/default.jsp, with no leading '/', and the inbound request URL is http://host:port/context\_root/folder or http://host:port/context\_root/folder/, the request is redirected to http://host:port/context\_root/folder/folder/default.jsp, which is relative to the current request URI.

This behavior is found in the Servlet 3.0 feature when the `com.ibm.ws.webcontainer.redirectwithpathinfo` property is set to **true**. This property is ignored in the Servlet 3.1 feature and the behavior defaults, as described.

## Default error pages

The IBM extended function is the ability to specify a default error page with a web extension, such as `ibm-web-ext.xml`.

As a function of Servlet 3.0 and higher, default error pages are a modification of the ability to specify error pages. As with normal (non-default) error pages, default error pages are specified in web module descriptors (`web.xml`), and in web fragment descriptors (`web-fragment.xml`).

Normal (non-default) error pages specify either an exception-type or an error-code. A default error page omits both exception-type and error-code. A default error page is used when a servlet throws an exception or sets an error-code result and no configured error page matches the type of the exception or matches the set error code.

The ability to define default error pages is provided by the Servlet 3.0 specification, and is supported by the Servlet 3.0 schemas. Default error pages are error pages that do not contain an **exception-type** or an **error-code** element, according to the Servlet 3.1 specification.

Examples of error pages and default error pages follow.

### Default error page precedence rules

Three rules apply for determining precedence for default error pages in the `web.xml`, `web-fragment.xml`, and `ibm-web-ext.xml` files.

- Rule 1: `web.xml` and `web-fragment.xml` files.

When a default error page is specified in the `web.xml` file, it overrides (masks) any default error page that is specified in a `web-fragment.xml` file. Also, there is no error if, in addition, multiple `web-fragment.xml` files specify default error pages.

- Rule 2: `web-fragment.xml` and `web-fragment.xml`.

When a default error page is not specified in the `web.xml` file, an error condition exists if different default error pages are specified by two or more `web-fragment.xml` files.

- Rule 3: `ibm-web-ext.xml` and `web.xml` or `web-fragment.xml` files.

The rule of precedence between the `ibm-web-ext.xml` file and either the `web.xml` or `web-fragment.xml` files depends on the web container feature level.

When the web container feature level is 3.0, a default error page that is defined by an `ibm-web-ext.xml` file has precedence over a default error page that is defined in `web.xml` or `web-fragment.xml` files.

**Note:** When the web container uses feature level 3.0, you cannot use Servlet 3.1 schemas. Refer to the rule on using default error pages for servlet 3.0 schemas.

When the web container feature level is 3.1 or higher, a default error page that is specified by `web.xml` or `web-fragment.xml` file has precedence over a default error page that is specified in the `ibm-web-ext.xml` file.

## Schema rules

Two rules apply for whether default error pages are processed in `web.xml` or `web-fragment.xml` files. The rules depend on the web container feature version, which Servlet schema is in use, and the setting of a Java custom property.

These rules arose because IBM WebSphere Application Server traditional V8.0, did not support default error pages in the V8.0 general availability release. Support for default error pages was added to WebSphere Application Server traditional, in a service pack, by APAR PM94199.

Support for default error pages was added to WebSphere Application Server Liberty, in a service pack, by APAR PI05845. Because these updates are a change of externally visible function, the new function is disabled by default and must be enabled by a Java system property.

- Rule 1: Default error pages using Servlet 3.0 schema and using web container feature version 3.0.

When the web container feature version is 3.0 and a default error page is specified in `web.xml` or `web-fragment.xml` files that use a Servlet 3.0 schema, the default error pages are processed only if `com.ibm.ws.webcontainer.allowdefaulterrorpage` Java system property is set to **true**. When the Java system property is not set, or is not set to **true**, the default error page is ignored. A default error page that is specified by the `ibm-web-ext.xml` file is used.

- Case 2: Default error pages by using web container feature version 3.1.

When the web container feature version is 3.1 or higher, a default error page that is specified in the `web.xml` file or the `web-fragment.xml` file is always processed, regardless of what servlet schema version is used, and regardless of whether the Java custom property is set.

This case occurs when a descriptor uses a Servlet 3.1 schema, since the processing of a descriptor that uses a Servlet 3.1 schema requires web container feature version 3.1.

**Attention:** Web fragments were not added until Servlet 3.0. The `web-fragment.xml` file has no schema from Servlet 2.5.

## Error page and default error page examples

A default error page that is defined in an `ibm-web-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext xmlns="http://websphere.ibm.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1
 version="1.0">

 <default-error-page uri="/ExtErrorPage.html"/>
</web-ext>
```

An error-code error page element, which is defined in either the `web.xml` file or `web-fragment.xml` file:

```
<error-page>
 <error-code>404</error-code>
 <location>/ErrorCodeErrorPage.html</location>
</error-page>
```

An exception-type error page element, which is defined in either the `web.xml` file or `web-fragment.xml` file:

```
<error-page>
 <exception-type>javax.servlet.ServletException</exception-type>
 <location>/ExceptionTypeErrorPage.html</location>
</error-page>
```

A default error page element, which is defined in either the `web.xml` file or `web-fragment.xml` file:



```
<error-page>
 <location>/DefaultErrorPage.html</location>
</error-page>
```

### Schema examples

Example header of a web.xml file that uses a Servlet 2.5 schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 version="2.5">
```

Example header of a web.xml file that uses a Servlet 3.0 schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0">
```

Example header of a web.xml file that uses a Servlet 3.1 schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
 version="3.1">
```

Example header of a web-fragment.xml file that uses a Servlet 3.0 schema:

```
<?xml version="1.0" encoding="utf-8"?>
<web-fragment xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-fragment_3_0.xsd"
 version="3.0">
```

Example header of a web-fragment.xml file that uses a Servlet 3.1 schema:

```
<?xml version="1.0" encoding="utf-8"?>
<web-fragment xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-fragment_3_1.xsd"
 version="3.1">
```

### Servlet 3.1 feature functions:

The product supports a subset of Servlet 3.1 functions. View the clarifications and descriptions of some of the functions available.

Descriptions of the new Servlet 3.1 functions are provided in the specification and are not described here. However, additional considerations for the Servlet 3.1 feature are as follows:

#### Asynchronous I/O

A new feature of the Servlet 3.1 feature specifies that when the non-blocking read is started, any resource during the remaining request lifetime cannot call APIs, which can result in a blocking read. For example, for a POST request after the read listener is set by the resource, any subsequent call to `getParameter()` and `getPart()` API results in an `IllegalStateException`.

You must consider setting timeout with the `AsyncContext.setTimeout` API when you work with async servlets, otherwise the container default value (for example, 30 seconds) is used. The timeout resets each time async starts using the `ServletRequest.startAsync` API is called and expires when the `AsyncContext.complete` API is not called within the timeout period that follows the last time async started. When you use the async I/O support that is provided by the Servlet 3.1 feature, set the timeout

value with the `AsyncContext.setTimeout` API to also allow for async I/O to complete. Completion depends on other external factors, such as environment or network speed.

## Upgrade processing

| **Important:** Use the `ServletOutputStream` class with the `WriteListener` interface and the  
| `ServletInputStream` class with the `ReadListener` interface. Do not use these classes with the  
| `ObjectInputStream` class or the `ObjectOutputStream` class. These classes circumvent some of the required  
| checks for the `ReadListener` and `WriteListener` interfaces, mainly the `isReady` checks, and can cause  
| unexpected behavior.

| Upgrade processing is a Servlet 3.1 feature that has non-blocking read and write capability. When the read or write operations are async, there are no limits on how much time the server waits for the operation to complete. You can set the timeouts with the web container custom properties in the `server.xml` file, such as `upgradereadtimeout` and `upgradewritetimeout`. See the following example of a timeout of 5 seconds:

```
<webContainer upgradereadtimeout="5000" />
<webContainer upgradewritetimeout="5000" />
```

The request must not be upgraded by using the upgrade feature for Servlet 3.1 when the request is being handled by async servlet.

The application that supports the Servlet 3.1 feature for upgrade requires that the connection on the request remains open between the client and the application that hosts the upgrade. If the application does not initiate the `WebConnection.close()` when the upgrade processing is complete from its handler or any other resources, such as `ReadListener` or `WriteListener`, the TCP connection remains open until the server recycles.

When you use an `UpgradeHandler` and a `ReadListener` from the Servlet 3.1 feature, the `ReadListener.onAllDataRead` method is invoked only when the client closes the connection to the server that hosts the upgraded application. The Javadoc for `onReadListener.onAllDataRead` returns the following message:

Invoked when all data for the current request is read.

In the Upgrade case, the server does not know the end of the data because upgraded data is not delimited the way that HTTP request body data is. Aside from when the client connection closes, there is no determination for the end of the data.

## Form based authentication

After successful authentication, a client is redirected to the resource of the original request. The Servlet 3.1 specification specifies: To improve the predictability of the HTTP method of the redirected request, containers should redirect using the 303 (SC\_SEE\_OTHER) status code, except where interoperability with HTTP 1.0 user agents is required; in which cases the 302 status code should be used. The Servlet-3.1 feature maintains interoperability with HTTP 1.0 user agents and always uses the 302 status code. For more information on configuring Servlet 3.1 for security, read the [Configuring Liberty for Servlet 3.1](#) topic.

## Large post data

The addition of the `ServletRequest.getContentLengthLong()` API requires support for receiving post data of a length greater than `Integer.MAX_VALUE` and cannot be fully accommodated in a single-byte array or string.

This addition has implications when you obtain post data content that use APIs that return content in a string or `byte[]`. For example, the `javax.servlet.ServletRequest` methods for accessing parameters:

```
String getParameter(String name)
String[] getParameterValues()
Map<String,String> getParameterMap()
```

It is possible to send post data that contains multiple parameters, which when combined, have a length of greater than `Integer.MAX_VALUE`. However, each individual parameter name and parameter value must be less than `Integer.MAX_VALUE` in length.

Sending a large amount of post data include these additional considerations:

- You must send post data in chunks of less than `Integer.MAX-VALUE` in length.
- Post data that is processed by the web container, such as parameters or parts, must be fully read before processing starts. The post data might impose significant memory requirements for large post data because it might require as much memory as double the size of the post data in order for web container processing to be successful.

## Configuring Liberty for Expression Language 3.0



8.5.5.5

You can configure Liberty for the Expression Language (EL) 3.0 feature, which provides full support for the EL 3.0 specification.

### About this task

To configure a Liberty server to run an application that is enabled for EL 3.0, you must set the `<el-3.0>` feature.


### Procedure

Update the `server.xml` file to add the `<el-3.0>` feature. For example:

```
<featureManager>
 <feature>el-3.0</feature>
</featureManager>
```

### Important:

- The EL 3.0 feature does not require any additional features. You can configure it independently of JavaServer Pages (JSP) 2.3.
- The JSP 2.3 feature requires the EL 3.0 feature. When you configure the JSP 2.3 feature, the EL 3.0 feature also loads into the server run time.
- You can use other Java EE 6 features, such as JSF 2.0 and CDI 1.0, with the EL 3.0 feature.
- You can choose between EL 3.0 and EL 2.2 (included in the JSP 2.2 feature) for each server instance, but you must consider any behavior changes. If the required behavior is contained only in the EL 3.0 feature, then you must use the EL 3.0 feature. If an existing application would be adversely impacted by behavior changes in the EL 3.0 feature, then use the EL 2.2 feature (included in JSP 2.2) to preserve the existing behavior for that application.
- It is not possible to use both the EL 3.0 feature and JSP 2.2 (includes EL 2.2) features in the same Liberty server. If both features are configured, it produces an error:  
CWWKF0033E: The singleton features com.ibm.websphere.appserver.javax.el-2.2 and com.ibm.websphere.appserver.javax.el-3

See  8.5.5.5 “Expression Language 3.0 feature functions” on page 1068 to learn about changes in the EL 3.0 feature, compared to the EL 2.2 feature.

### Results

The EL 3.0 feature is enabled and loads in the Liberty server at run time.

## Expression Language 3.0 feature functions: 8.5.5.5

The Expression Language (EL) 3.0 feature provides full support for the EL 3.0 specification.

Descriptions of EL 3.0 functions are provided in the EL 3.0 specification and are not fully described here. However, some of the key enhancements include the following:

- EL 3.0 is now available as a separate feature and you can configure it independently of JavaServer Pages (JSP) 2.3.
- Add support for Lambda expressions (value expression with parameters). For more information, see Section 1.20 of the EL 3.0 specification.
- Addition of operations on collections objects. For more information, see Chapter 2.0 of the EL 3.0 specification.
- New operators:
  - String concatenation. For more information, see section 1.8 of the EL 3.0 specification.
  - Assignment. For more information, see section 1.13 of the EL 3.0 specification.
  - Semi-colon. For more information, see section 1.14 of the EL 3.0 specification.
  - Field and Methods. For more information, see Section 1.22 of the EL 3.0 specification.

**Important:** There is a change in the EL 3.0 feature that might break existing applications. The default coercion for nulls to non-primitive types (except String) returns nulls. For example, a null that is coerced to a Double now returns a null value, whereas before it returned 0.0. The following code example describes this scenario:

```
Integer number=null;
factory.coerceToType(number, java.lang.Double.class)
```

## Configuring Liberty for JavaServer Faces 2.2

8.5.5.6

You can configure Liberty for the JavaServer Faces (JSF) 2.2 feature, which provides full support for the JSF 2.2 specification.

### About this task

The Liberty JSF implementation is based on the MyFaces open source implementation. To configure a Liberty server to run an application that is enabled for JSF 2.2, you must set the <jsf-2.2> feature.

### Procedure

Update the server.xml file to add the <jsf-2.2> feature. For example:

```
<featureManager>
 <feature>jsf-2.2</feature>
</featureManager>
```

### Important:

Consider the following points when you use JavaServer Faces 2.2:

- JSF 2.2 feature does not implicitly load the bean validation feature like the JSF 2.0 feature does. When you migrate your application from JSF 2.0 to JSF 2.2, and your application uses bean validation, you must also enable the beanValidation-1.1 feature.
- The JSF 2.2 feature requires the servlet-3.1, jsp-2.3, timedexit-1.0, and el-3.0 features. When the JSF 2.2 feature is enabled in the server.xml file, each of these features is also enabled.
- You cannot run the JSF 2.2 feature with Java EE 6 features; for example, servlet-3.0, jsp-2.2 and cdi-1.0.

- You can choose between the JSF 2.0 and JSF 2.2 feature implementations for each server instance, but you must consider any behavior changes. If the required behavior is contained only in the JSF 2.2 feature, then you must use the JSF 2.2 feature. If an existing application would be adversely impacted by behavior changes in the JSF 2.2 feature, then use the JSF 2.0 feature to preserve the existing behavior for that application.
- It is not possible to use both the JSF 2.0 and JSF 2.2 features in the same Liberty server. If both features are configured, it produces an error:  
CWWKF0033E: The singleton features jsf-2.0 and jsf-2.2 cannot be loaded at the same time. The configured features jsf
- JSF 2.2 is compatible with earlier releases, such as JSF 2.1 and JSF 2.0; however, consider the following exceptions:
  - An error in previous versions of the JSF specification caused exceptions to be swallowed that now are propagated to the exception handler. Read Backward Compatibility with Previous Versions in the overview section of the JSF 2.2 specification.
  - Changes made to the specification for Composite Component Attribute ELResolver and Composite Computer Metadata. Read Backward Compatibility with Previous Versions in the overview section of the JSF 2.2 specification.

## Results

The JSF 2.2 feature is enabled and loads in the Liberty server at run time.

## What to do next

To use the FlowBuilder API to create Flows with FlowBuilder annotations, it is required that the relevant CDI producer method is declared within a managed bean or a session bean class. To ensure that the class is managed correctly by CDI, define the producer method class as a managed bean (by giving it a scope), or set CDI bean-discovery-mode to all. You can set the CDI bean-discovery-mode to all in the beans.xml file in your web archive:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
 bean-discovery-mode="all">
</beans>
```

## JavaServer Faces 2.2 feature functions: 8.5.5.6

The JavaServer Faces (JSF) 2.2 feature provides full support for the JSF 2.2 specification.

Descriptions of JSF 2.2 functions are provided in the JSF 2.2 specification and are not fully described here. However, some of the key enhancements are as follows:

- Faces Flows enables developers to logically group JSF views to represent modules of functions. Each module has a well-defined set of entry and exit points. Combining modules creates a flow of functions. An example of a flow of functions is an order checkout process. Read section 7.5 of the JSF 2.2 specification.
- Resource Library Contracts enable resource libraries to reside in the contracts directory of the web-app root directory, or in the META-INF/contracts entry name in the JAR file. Read section 10.1.3 of the JSF 2.2 specification.
- JSF 2.2 introduced the ability for applications to use stateless views. Read section 7.8.1.1 of the JSF 2.2 specification.
- JSF 2.2 can handle HTML5 attributes through pass-through elements and pass-through attributes. Read section 10.1.4 of the JSF 2.2 specification

## Configuring Liberty for JavaServer Pages 2.3



8.5.5.5

You can configure Liberty for the JavaServer Pages (JSP) 2.3 feature, which provides full support for the JSP 2.3 specification.

### About this task

To configure a Liberty server to run an application that is enabled for JSP 2.3, you must set the `<jsp-2.3>` feature.

### Procedure

Update the `server.xml` file to add the `<jsp-2.3>` feature. For example:

```
<featureManager>
 <feature>jsp-2.3</feature>
</featureManager>
```

### Important:

- The `jsp-2.3` feature requires both the `servlet-3.1` and `el-3.0` features and when configured, causes these features to load.
- You cannot use `jsp-2.3` with the `servlet-3.0` feature.
- You can use other Java EE 6 features, such as JSF 2.0 and CDI 1.0, with the JSP 2.3 feature.
- You can choose between the JSP 2.2 and JSP 2.3 feature implementations for each server instance, but you must consider any behavior changes. If the required behavior is contained only in the JSP 2.3 feature, then you must use the JSP 2.3 feature. If an existing application would be adversely impacted by behavior changes in the JSP 2.3 feature, then use the JSP 2.2 feature to preserve the existing behavior for that application.
- It is not possible to use both the JSP 2.2 and JSP 2.3 features in the same Liberty server. If both features are configured, it produces an error:

CWWKF0033E: The singleton features jsp-2.3 and jsp-2.2 cannot be loaded at the same time. The configured features jsp-2.3



8.5.5.5

“JavaServer Pages 2.3 feature functions” to learn about changes in the JSP 2.3 feature.

### Results

The JSP 2.3 feature is enabled and loads in the Liberty server at run time.

### JavaServer Pages 2.3 feature functions:



8.5.5.5

The JavaServer Pages (JSP) 2.3 feature provides full support for the JSP 2.3 specification.

Descriptions of the new JSP 2.3 functions are provided in the JSP 2.3 specification and are not described here. However, the new specification additions are minor, as compared to the JSP 2.2 feature:

- Support for Expression Language (EL) 3.0
- Availability of Servlet 3.1 APIs to a JSP.
- There are no known behavior differences between JSP 2.2 and JSP 2.3 that directly prevent a JSP that runs with JSP 2.2 from successfully running with JSP 2.3. However, if a JSP uses Expression Language (EL) or Servlet API functions, you must consider the changes between Servlet 3.1 and Servlet 3.0, and between EL 3.0 and EL 2.2.




# Administering Contexts and Dependency Injection applications on Liberty



8.5.5.6

Liberty provides support for contexts and dependency injection in applications using the Liberty features `cdi-1.0` and `cdi-1.2`.

## Procedure

-  [“Configuring Liberty for Contexts and Dependency Injection 1.2”](#)
-  [“Contexts and Dependency Injection 1.2 overview”](#)
-  [“Contexts and Dependency Injection 1.2 behavior changes”](#) on page 1072

## Configuring Liberty for Contexts and Dependency Injection 1.2



8.5.5.6

You can configure Liberty for the Contexts and Dependency Injection (CDI) 1.2 feature, which provides full support for the Contexts and Dependency Injection 1.2 specification.

### About this task

To configure the Liberty server to run an application that is enabled for CDI 1.2, you must set the `<cdi-1.2>` feature.

## Procedure

Update the `server.xml` file to add the `<cdi-1.2>` feature. For example:

```
<featureManger>
 <feature>cdi-1.2</feature>
</featureManger>
```

### Note:

- You can use other Java EE 7 features, such as `jsp-2.3` and `jsf-2.2` with the `cdi-1.2` feature. However, you cannot use Java EE 6 features such as `jsp-2.2` and `jsf-2.0` with the `cdi-1.2` feature.
- You can choose between the CDI 1.0 and CDI 1.2 feature implementations for each server instance, but you must consider the behavior changes. If the behavior is only contained in the CDI 1.2 feature, then you must use the CDI 1.2 feature. If an existing application might be adversely impacted by behavior changes in the CDI 1.2 feature, then use the CDI 1.0 feature to preserve the existing behavior for that application.
- It is not possible to use both the CDI 1.0 and CDI 1.2 features in the same Liberty server. If both features are configured, it produces an error. Read the CDI 1.2 behavior changes topic to learn about changes from CDI 1.0 to CDI 1.2.

## Results

The CDI 1.2 feature is enabled and loads in the Liberty server at run time.

## Contexts and Dependency Injection 1.2 overview



8.5.5.6

Liberty provides support for contexts and dependency injection in applications by using the Liberty features `cdi-1.0` and `cdi-1.2`.

The Contexts and Dependency Injection (CDI) 1.2 feature provides full support for the CDI 1.2 specification. Full descriptions of the CDI 1.2 functions are provided in the CDI 1.2 specification, see Contexts and Dependency Injection for the Java EE platform.

The set of services that are provided by the CDI 1.2 feature includes a well-defined lifecycle for stateful objects that are bound to lifecycle contexts and a typesafe dependency injection mechanism.

### **Using Contexts and Dependency Injection 1.2 with JavaServer Faces applications**

You can use the CDI 1.2 feature with the JavaServer Faces (JSF) 2.2 feature to enable JSF applications to take advantage of the sophisticated context and dependency injection model that is provided in the CDI 1.2 feature. This service is provided through integration with the Unified Expression Language (EL) which enables any contextual object to be used directly within a JSF or JavaServer Pages (JSP) page.

### **Using Contexts and Dependency Injection 1.2 with Enterprise JavaBeans (EJB)**

You can use the CDI 1.2 feature with the Enterprise JavaBeans (EJB) 3.2 feature to enhance the EJB component model with contextual lifecycle management. The services that are provided by the CDI 1.2 feature integrate the Java EE web tier with Java EE enterprise services. In particular, this enables EJB components to be used as JSF managed beans, thus integrating the programming models of EJB and JSF.

### **Using Contexts and Dependency Injection 1.2 with Servlet 3.1**

You can use the CDI 1.2 feature with the Servlet 3.1 feature to enable servlet applications to take full advantage of the services that are provided by the CDI 1.2 feature. Using both features enables contextual managed beans to be injected into servlet applications by using field, method or constructor injection. The CDI 1.2 feature also provides automatic registration of servlet listeners, filters, and interceptors.

### **Java Interceptors in Contexts and Dependency Injection 1.2 Applications**

The CDI 1.2 feature extends the Java model for interceptors. The CDI 1.2 feature provides the ability to associate interceptors with beans. The interceptors are bound by using typesafe interceptor bindings. This model can be extended to EJB beans when both the CDI 1.2 and EJB 3.2 features are loaded onto the Liberty server.

### **Contexts and Dependency Injection 1.2 behavior changes**



8.5.5.6

The Contexts and Dependency Injection (CDI) 1.2 implementation contains some behavior changes that might cause an application that was migrated from CDI 1.0 to behave differently or fail on CDI 1.2.

You can choose between the CDI 1.0 and CDI 1.2 feature implementations for each server instance, with consideration for behavior changes. If the required behavior is contained in the CDI 1.2 feature only, then you must use the CDI 1.2 feature. If an existing application might be adversely impacted by behavior changes in the CDI 1.2 feature, then by using the CDI 1.0 feature the existing behavior is preserved for that application. It is not possible to use both the CDI 1.0 and CDI 1.2 features in the same server as the features are not compatible. If you configure both features, the server produces a configuration error.

The CDI 1.0 feature is built on Apache OpenWebBeans implementation of CDI. The CDI 1.2 feature is built on the Weld implementation of CDI. The behavior changes introduced are due to the differences in the two implementations.



## The conversation ID CID

In the CDI 1.0 implementation, the CID is globally unique. In CDI 1.2, it is unique per HTTP session. This behavior is in line with the CDI specification and is a convention that is chosen by the Weld. To get a globally unique CID, the CID must be specified at conversation start by calling `Conversation.begin`.

## Referencing schemas in the beans.xml file

In a CDI 1.2 implementation, here is an example of a schema that is referenced in the `beans.xml` file:

```
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
```

If you use an invalid schema, the server gives an exception error. You can turn off the validation of the `beans.xml` file by setting the `org.jboss.weld.xml.disableValidating=true` JVM property, which also prevents the error from being produced. If the `beans.xml` file specifies decorators or interceptors, a valid schema must be used, otherwise the decorators and interceptors are not correctly instantiated.

## Implicit bean archives

The CDI 1.2 implementation defines two different types of bean archives: explicit and implicit.

An explicit bean archive is an archive that contains a `beans.xml` file:

- with a version number of 1.1 (or later), and with the `bean-discovery-mode` of `all`
- with no version number
- that is an empty file

An implicit bean archive is any other archive, which contains one or more bean classes with a bean defining annotation as defined in the specification in Section 2.5.1, "Bean defining annotations", or one or more session beans. See the specification, Contexts and Dependency Injection for the Java EE platform.

When you update the schema to a CDI 1.2 implementation, to keep the bean archive as explicit the bean discovery mode must be set to `all`:

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
bean-discovery-mode="all"
version="1.1">
```

**Note:** An implicit bean archive discovers only beans that have a bean defining annotation.

This new type of bean archive can result in an archive that is not intended to be a CDI bean archive but it becomes an implicit bean archive in the CDI 1.2 implementation. To stop this behavior, you can add a `beans.xml` file with the bean discovery mode set to `none`, preventing the archive from being a bean archive. An alternative solution is to add the following property to the `server.xml` file for your Liberty server:

```
<cdi12 enableImplicitBeanArchives="false"/>
```

Setting this property to `false` prevents the archives without `beans.xml` files from becoming implicit bean archives.

Setting this property to `false` gives an improved performance at startup time.

## Administering JavaMail on Liberty



8.5.5.6

You can configure JavaMail on Liberty by adding and configuring elements in the `server.xml` file.

### About this task

If you have an external mail server, you can use the JavaMail API to send and receive email on applications that are running on a Liberty server. The API allows applications to interact with the external mail server by providing common store and transport protocols, such as POP3, IMAP, and SMTP.

Liberty supports JavaMail 1.5. For more information about JavaMail 1.5, see the JavaMail API documentation.

For information about the elements and attributes that you can use to configure JavaMail on Liberty, see “JavaMail 1.5” on page 548.

### Procedure

1. In the `server.xml` file, add the `javaMail-1.5` feature. After you add the feature, you can call the JavaMail libraries in any application that runs on the server.

```
<featureManager>
 <feature>javaMail-1.5</feature>
</featureManager>
```
2. Optional: If you want to create a `javax.mail.Session` object, add and configure a `mailSession` element. After the mail session is configured, the session is created and injected by using the Java Naming and Directory Interface (JNDI).

**Note:** If you use the standard JNDI context, `java:comp/env/mail/exampleMailSession`, configure the `jndiName` attribute as `jndiName="mail/exampleMailSession"`.

```
<mailSession mailSessionID="examplePop3MailSession"
 jndiName="ExampleApp/POP3Servlet/exampleMailSession"
 description="POP3 javax.mail.Session"
 storeProtocol="pop3"
 transportProtocol="smtp"
 host="exampleserver.com"
 user="jamaexample@example.com"
 password="example"
 from="smtp@testserver.com">
 <property name="mail.pop3.host" value="pop3.example.com" />
 <property name="mail.pop3.port" value="3110" />
</mailSession>
```

---

## Administering Liberty using Admin Center



8.5.5.2


You can use WebSphere Liberty Administrative Center ("Admin Center") to administer Liberty servers and applications and other resources from a web browser on a smartphone, tablet, or computer.

### About this task








Use the Admin Center with a Liberty server or Liberty Core installation. An installation of the WebSphere Application Server full profile is not required.

Admin Center offers the ability to start, stop, and view details about Liberty servers and applications. It also offers the ability to edit server configuration files, to view bookmarked information, to add tools, to monitor server resources, and to deploy server packages on hosts within a collective. Advantages of the Admin Center design include:

- A user interface with a mobile look-and-feel for web browsers on a smartphone or tablet
- Support for multiple lightweight, task-oriented tools
- A toolbox that you can customize by selecting tools from a catalog, specifying bookmarks, or specifying user preferences

 **Watch:** The Touring Admin Center video shows the user interface and briefly describes Admin Center features. [Transcript]

## Procedure

1. Set up Admin Center.
  - Install Liberty with Admin Center.
  - Configure the server.xml file to enable Admin Center and secure login.
  - If the server is not running, start the server.
2. Log in to Admin Center.
  - Point a web browser at Admin Center.
  - On the Welcome page, specify an authorized user name and password to log in.
3. Use the Toolbox.
  - Select a tool or bookmark.
  - Add a tool.
  - Remove a tool.
  - Add or remove a bookmark.
  -  Rearrange tool and bookmark icons.
  - Filter toolbox contents.
  - Edit user preferences such as text direction.
  -  Uninstall an Admin Center tool.
4.  Use the Server Config tool to view and edit server configuration files in the Liberty topology.
  - Enable editing of files to change server.xml or any other files in the server configuration directory.
  - View or edit a configuration file in the Server Config tool.
  - Customize a default setting for Design or Source mode.
5. Use the Explore tool to explore and manage resources in the Liberty topology.
  - View a summary of all resources on the Explore tool Dashboard.
  - View a summary of all applications, clusters, servers, or hosts.
  - View details about a resource.
  -  View details about all servers on a Liberty runtime.
  - Start, stop, or restart a resource.
  -  Search for resources.
6.  Optional: View charts of server or application metrics in the Monitor view of the Explore tool.
  - Open the Monitor view on a server or application.
  - Show or hide charts.
  - Show or hide chart legends.
  -  View chart data in a table.
7. After completing work in Admin Center, log out.

## Setting up Admin Center



 8.5.5.2

Admin Center is a web user interface that runs on Liberty V8.5.5.2 and later servers. After installing Liberty and creating a server, configure the `server.xml` file.

## Before you begin

Install WebSphere Application Server Liberty with Liberty Administrative Center ("Admin Center"). The Installing Liberty Repository assets topic lists the ways to install assets such as Admin Center. The quickest way to install Admin Center is to run the `installUtility` command or the `featureManager` command:

1. If you have not done so already, install WebSphere Application Server Liberty V8.5.5.2 or later.

**Restriction:** Ensure that you use a Java virtual machine (JVM) that supports Liberty products and Secure Sockets Layer (SSL). Do not use an IBM JVM available with a WebSphere Application Server traditional product, such as Network Deployment, for your Liberty installation with Admin Center. By default, the IBM JVM available with a traditional product points to security classes that are available only with a traditional product, and not to security classes needed by Admin Center. Using an IBM JVM available with a traditional product can cause Admin Center to not display in a browser.

2. Open a command window at the main directory of the Liberty installation. For example, open a command window at `c:\wlp`.
3. Run a command to install the `adminCenter-1.0` feature.

**8.5.5.6** For Version 8.5.5.6 or later, run the `installUtility` command:

```
bin/installUtility install adminCenter-1.0
```

For Version 8.5.5.5 or earlier, run the `featureManager` command:

```
bin/featureManager install adminCenter-1.0 --when-file-exists=ignore
```

4. For the Liberty Developers edition, run a command to install the `collectiveController-1.0` feature.

**8.5.5.6** For Version 8.5.5.6 or later, run the `installUtility` command:

```
bin/installUtility install collectiveController-1.0
```

For Version 8.5.5.5 or earlier, run the `featureManager` command:

```
bin/featureManager install collectiveController-1.0 --when-file-exists=ignore
```

This step applies only to the Developers edition and not to the Network Deployment, z/OS or Core editions of Liberty. Optionally, you can install the `collectiveController-1.0` feature before installing the `adminCenter-1.0` feature.

For more information, go to the WASdev website, select the **Downloads** tab, and search the Liberty repository for the Admin Center asset.

To install Admin Center on hosts that cannot access the internet-based Liberty repository, first install Liberty and the Admin Center feature on a host that can access the internet. Then transfer the installation to the target hosts. For information about packaging Liberty servers and runtimes for deployment to other hosts, see Packaging a Liberty server by using developer tools and Packaging a Liberty server from the command line.

## About this task

You can set up Admin Center on stand-alone servers and on collective controllers. This topic focuses on setting up a stand-alone Liberty server.

To enable Admin Center on a collective controller, see Configuring a Liberty collective and the example in Deploying resources with Admin Center. Ensure the `server.xml` file of the collective controller includes `<feature>adminCenter-1.0</feature>` in the feature manager configuration and sets a host value in the `httpEndpoint` element, such as `host="*"` so all hosts can access the collective controller.

## Procedure

1. If your Liberty installation does not have a server, create a Liberty server.

For example, in a command window at the `wlp/bin` directory, create a server named `myServer`.

```
server create myServer
```

The example command adds server files to the `wlp/usr/servers/myServer` directory.

2. Open an editor on the `server.xml` file of the Liberty server, and configure the server for Admin Center.

- a. Add the `adminCenter-1.0` feature to the feature manager.

```
<featureManager>
 <feature>adminCenter-1.0</feature>
</featureManager>
```

For more timely updates to server and application status in the Explore tool, also add the `websocket-1.1` or `websocket-1.0` feature to the server configuration.

```
<featureManager>
 <feature>adminCenter-1.0</feature>
 <feature>websocket-1.1</feature>
</featureManager>
```

WebSocket provides a live view of the topology regardless of size. Without the WebSocket feature, Admin Center periodically and frequently polls for changes.

- b. Add one or more users to configure a secure login. For example:

```
<quickStartSecurity userName="admin" userPassword="adminpwd" />
```

If user names or passwords include non-English characters, create the `jvm.options` file for the server and define the default client encoding as UTF-8:

```
-Ddefault.client.encoding=UTF-8
```

For information about the `jvm.options` file, see “Customizing the Liberty environment” on page 947.

- c. To protect keystore files that have server authentication credentials, define a keystore and assign it a password.

```
<keyStore id="defaultKeyStore" password="Liberty" />
```

For an example `server.xml` file that defines an Administrator and a non-Administrator and that defines a keystore, see the Example in this topic. For information about defining multiple administrative users, see “Setting up BasicRegistry and role mapping on Liberty” on page 1150.

- d. To access Admin Center from a smartphone, tablet, or remote computer, ensure that the `server.xml` file sets the `host` attribute of the `httpEndpoint` element to `*` (asterisk) or to a defined host name. By default, the `host` attribute is set to `localhost`.

```
<httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080"
 httpsPort="9443" />
```

- e. Save your changes to the `server.xml` file.

If you defined the default client encoding as UTF-8 for non-English characters in the `jvm.options` file and the user registry is in `quickStartSecurity` or `basicRegistry` elements, which store user names and passwords in the `server.xml` file, then save the `server.xml` file in UTF-8 encoding.

3. If the server is not running, start the server.

For example, in a command window at the `wlp/bin` directory, enter a **run** or **start** command.

```
server run myServer
```

Look for server messages that show the `adminCenter` web application is running. After Admin Center is running, you can point a web browser at the application and log in. See “Logging in to Admin Center” on page 1078.

 **Watch:** The Setting up Admin Center video demonstrates the procedure. [Transcript]

## Example: server.xml file that defines two authorized users

```
<server description="new server">

 <!-- Enable features -->
 <featureManager>
 <feature>adminCenter-1.0</feature>
 </featureManager>

 <!-- Define the host name for use by the collective.
 If the host name needs to be changed, the server should be
 removed from the collective and re-joined. -->
 <variable name="defaultHostName" value="localhost" />

 <!-- Define an Administrator and non-Administrator -->
 <basicRegistry id="basic">
 <user name="admin" password="adminpwd" />
 <user name="nonadmin" password="nonadminpwd" />
 </basicRegistry>

 <!-- Assign 'admin' to Administrator -->
 <administrator-role>
 <user>admin</user>
 </administrator-role>

 <keyStore id="defaultKeyStore" password="Liberty" />

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="9080"
 httpsPort="9443" />

</server>
```

## Logging in to Admin Center



8.5.5.2

After the Admin Center web application starts, you can point a web browser at Admin Center and log in. After logging in, the first page shown is the Toolbox.

### Before you begin

Ensure that you have a browser that supports Admin Center.

- For browsing the Admin Center on a computer: Firefox ESR 24, Firefox 29, Chrome 35, Safari 7, or Internet Explorer 11 browser (for best experience, use Chrome or Firefox browser)
- For browsing the Admin Center on an iPhone or iPad: Safari browser on iOS 6.x or 7.x
- For browsing the Admin Center on an Android mobile device: Chrome browser on Android 4.2 or 4.3

### Procedure

1. Point a web browser at Admin Center.

```
https://host_name:port_number/adminCenter/
```

For *host\_name*, specify the IP address or domain name server (DNS) host name of the computer on which the Liberty server is running. Specify `localhost` only if the browser is running on the same computer as the server.

For *port\_number*, specify the `httpsPort` value in the `server.xml` file. For example:

```
https://localhost:9443/adminCenter/
```

```
https://myhost.xyz.com:9443/adminCenter/
```

```
https://9.65.234.567:9443/adminCenter/
```

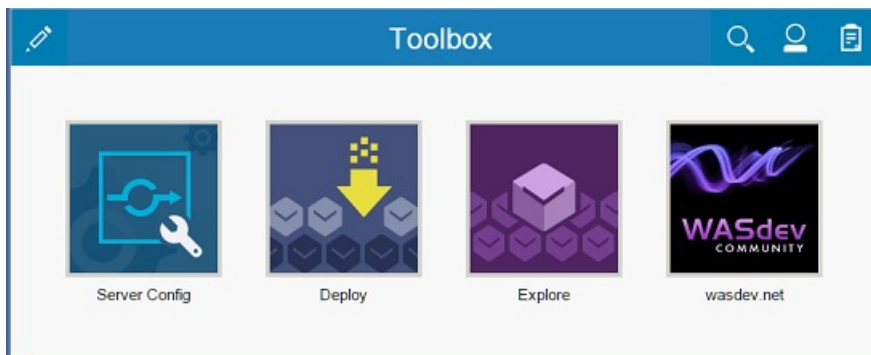
2. If your browser prompts you to confirm that the connection is trusted, specify an exception or otherwise enable the connection to continue to Admin Center.
3. Log in to Admin Center.

When you configured the `server.xml` file of the server to enable Admin Center, you defined one or more user names and passwords authorized to access the Admin Center. To log in to Admin Center, specify an authorized user name and password.

For example, specify the user name `admin` and the password `adminpwd`.

## What to do next

When you first access the Toolbox, it has the Server Config and Explore tools and a bookmark to `WASdev.net`. The Toolbox also has the Deploy tool if Admin Center is run on a collective controller. The Deploy tool is not available if Admin Center is run on a stand-alone server.



To add or remove tools and bookmarks, select  and use the Tool Catalog to customize your user-specific Toolbox.

**8.5.5.4** To view a tool, select its Toolbox icon or directly launch the tool using the URL that is shown in the browser when viewing the tool. For example, to directly launch the Explore tool, use a URL such as `https://localhost:9443/adminCenter/#explore`.

Later, to log out of Admin Center, select  > **Log out *user\_name***.

## Customizing the Toolbox

### **8.5.5.2**

When you first log in to Admin Center, the Toolbox contains the Server Config and Explore tools and a bookmark to `WASdev.net`. The Toolbox also has the Deploy tool if the server hosting Admin Center is a collective controller. You can edit the Toolbox contents by adding tools from the Tool Catalog or by adding bookmarks. A *tool* is a web application that performs a particular task and is typically provided by the product. A *bookmark* is a user-added link to any site. You can further customize your Toolbox by rearranging icons, filtering Toolbox contents, or editing preferences.

## About this task





Select tools, bookmarks, and menu choices in the Toolbox to use Admin Center. The Toolbox offers the following options:











- Select a tool or bookmark.
- Add a tool.
- Remove a tool.

- Add or remove a bookmark.
- Rearrange tool and bookmark icons.
- Filter toolbox contents.
- Edit user preferences such as text direction.
- **8.5.5.6** Uninstall an Admin Center tool.






## Procedure

- To work with an existing tool or to open a browser on a bookmarked URL, select the icon for the tool or bookmark.
- Add tools.
  1. Select  >  > **Add Tool**.
  2. Select  on the tool to add.
  3. Confirm that you want to add the tool.
  4. Select  to return to the Toolbox.

**8.5.5.4** After you add a tool, use the Toolbox to first launch the tool. To directly launch the tool at a later time, you can use the URL that is shown in the browser when viewing the tool.
- Remove tools.
  1. Select .
  2. Select  on the tool to remove.
  3. Confirm that you want to remove the tool.
  4. Select  to return to the Toolbox.
- Add or remove bookmarks.
  1. Select  >  > **Add Bookmark**.
  2. Enter the bookmark name and URL, and then select **Add** to add the bookmark to your Toolbox. To later remove the bookmark, select the bookmark and then .
  3. Select  to return to the Toolbox.
- **8.5.5.4** Rearrange tool and bookmark icons.
  1. Select .
  2. Drag and drop the icons into the order that you prefer.
  3. Select  to return to the Toolbox.
- Filter toolbox contents.
  1. Select .



2. Type the tool or bookmark name in the text field. Filtered search results are shown if the characters typed match the names of Toolbox contents.
  3. After searching, select  to close the text field.
- Edit your preferences.
    1. Select  > **Preferences**.
    2. For **Enable bidirectional support**, specify whether text displays left to right, the default, or displays right to left. Select **Contextual** for the first typed character to determine the direction. When the first character is in a language such as Arabic or Hebrew, control orientation and typing direction become right to left, with right alignment for text.
    3. Select  to return to the Toolbox.
  - **8.5.5.6** Uninstall an Admin Center tool.  
 By default, the Toolbox contains Admin Center tools such as the Explore tool and, if the server hosting Admin Center is a collective controller, the Deploy tool. You can use the feature manager to uninstall Admin Center tools to fully customize your Toolbox. Run the **uninstall** command for featureManager. For example, if you use DevOps you might want to uninstall the Deploy tool. To do so, run:
 

```
wlp/bin/featureManager uninstall com.ibm.websphere.appserver.adminCenter.tool.deploy-1.0
```

 Admin Center tools use the `com.ibm.websphere.appserver.adminCenter.tool.*` naming format:
    - `com.ibm.websphere.appserver.adminCenter.tool.explore-1.0.mf`
    - `com.ibm.websphere.appserver.adminCenter.tool.deploy-1.0`

## What to do next

Your selections are saved for the user name that was specified when logging in. If you later log in under the same user name, the Toolbox shows the same tool, bookmark, and preference selections as when you last logged out. If you log in under a different user name, the Toolbox shows the selections for that other user name.


## Editing server configuration files in Admin Center

**8.5.5.7**

You can use the Server Config tool to view and edit server configuration files in the Liberty topology. The Server Config tool displays configuration files such as a `server.xml` file in two modes. The Design mode displays the content of configuration files using graphical controls with inline documentation. The Source mode provides direct access to the file text and has content assist capabilities. You can customize the modes, for example, to add or remove parameter descriptions on the Design mode or to add or remove line numbers on the Source mode. Before you can edit files, you must add a `remoteFileAccess` element to the server configuration file; otherwise, files are shown in read-only mode.

**8.5.5.8**

### About this task

 **Watch:** The Getting started with the Server Configuration Tool for WebSphere Liberty video shows how to enable and use the tool.

### Procedure


1. Enable editing of files in the server configuration directory.
  - a. Open an editor on the server configuration file.  
 The configuration file for a server typically has a path name such as `wlp/usr/servers/server_name/server.xml`.
  - b. Add the following `remoteFileAccess` element to the server configuration file.

```
<remoteFileAccess>
 <writeDir>${server.config.dir}</writeDir>
</remoteFileAccess>
```

c. Save the file changes.

Server configuration files such as the server.xml and any included files no longer are read-only in the Server Config tool. You can change element and parameter settings, and otherwise edit the files.

2. View or edit a configuration file in the Server Config tool.

- From the Toolbox, select .
- If Admin Center is run on a collective controller, select the server that has the configuration file you want to view or edit.
- After a file is open for editing, you can switch between **Design** and **Source** modes.


In the Design mode, select elements in the configuration to see enabled features and parameter settings.

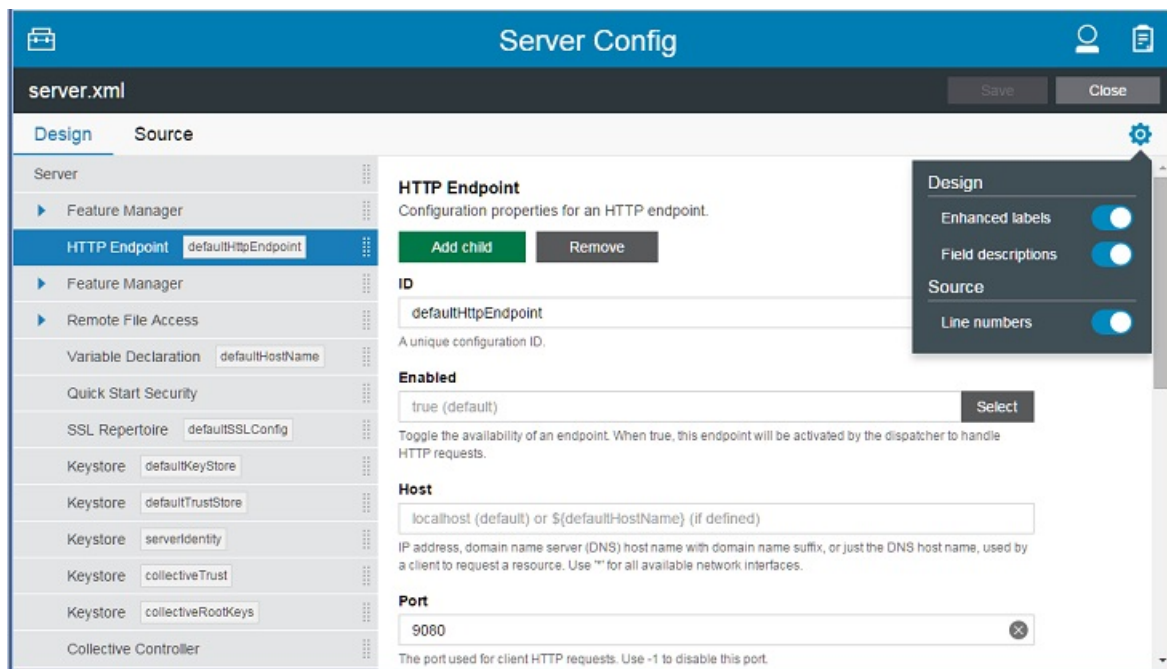
In the Source mode, hover on elements and parameters to view their documentation. You can press Ctrl+Space to use content assist, which helps you add new elements, parameters, and values.

- After you finish working with a file, select **Save** to save the file changes and then **Close** to return to the main page of the tool. To discard file changes, select **Close** and then confirm to not save the changes.

3. Optional: Customize the default settings for Design or Source mode.

By default, the Design mode uses enhanced labels for file elements. Enhanced labels display element names such as featureManager as Feature Manager. Also, the Design mode displays parameter descriptions by default. The Source mode displays line numbers by default. To change a default setting:

- Select .
- Enable or disable one or more default settings.
- Select **Close**.



## What to do next

To return to the Toolbox, select .

## Exploring and managing resources with Admin Center



You can use the Explore tool to view and manage resources in the Liberty topology.


### About this task

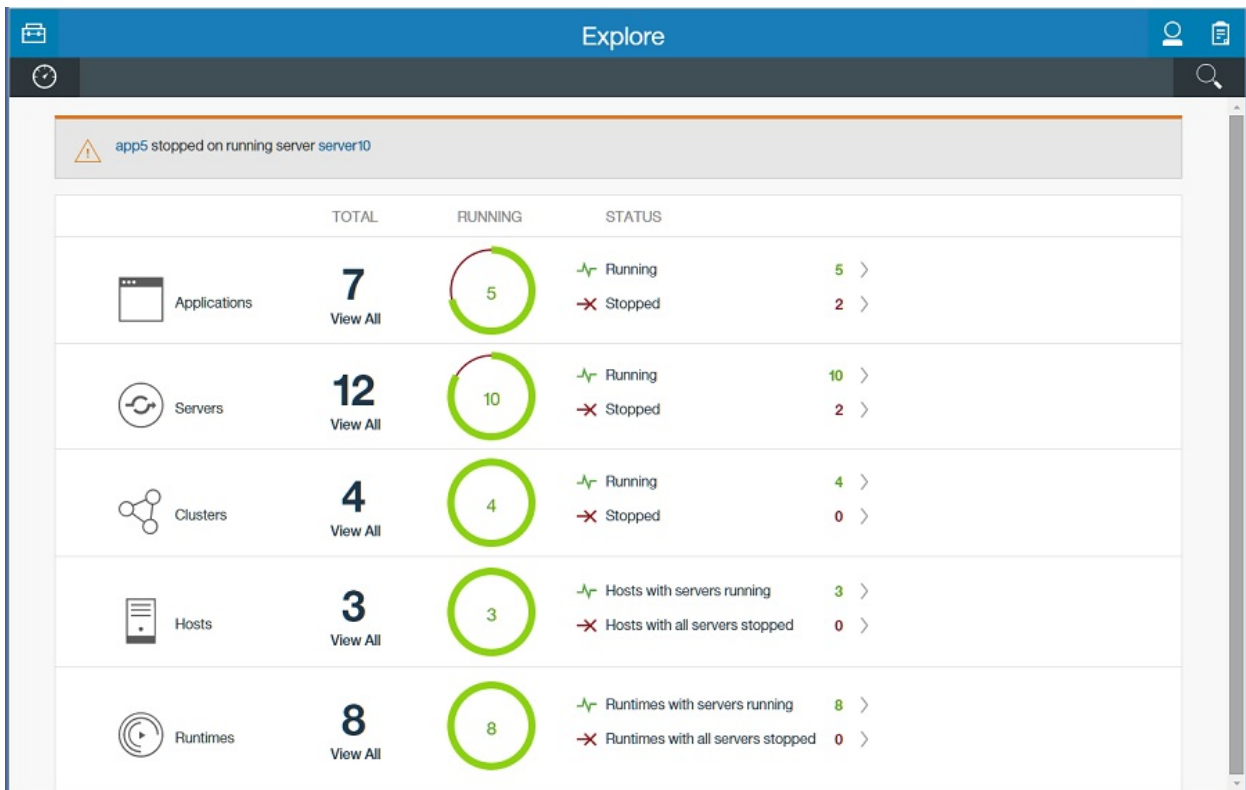
The Explore tool offers the following options to work with application, server, cluster, host, and runtime resources:

- View a summary of all resources on the Dashboard.
- View a summary of all applications, servers, clusters, hosts, or runtimes.
- View details about a resource.
- [8.5.5.4](#) View details about all servers on a Liberty runtime.
- Start, stop, or restart a resource.
- [8.5.5.5](#) Search for resources.

### Procedure

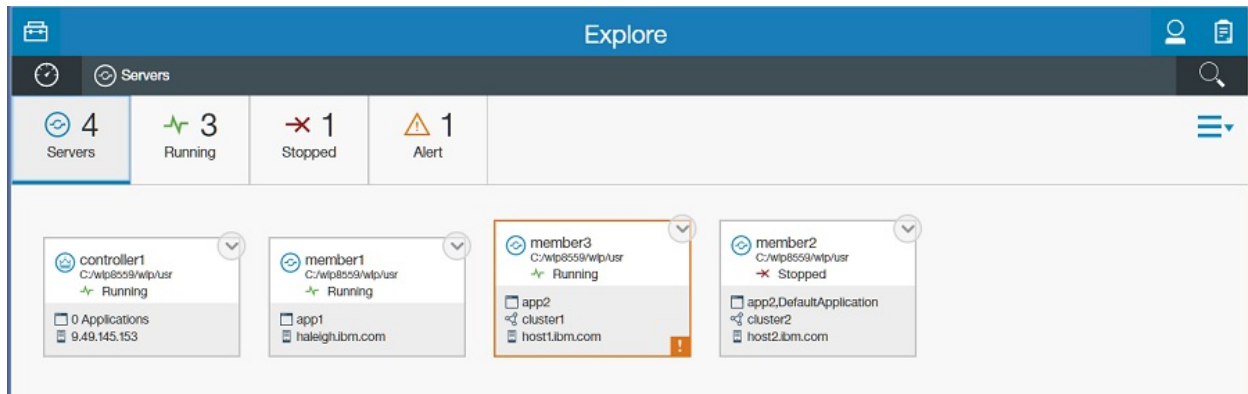
- View a summary of all resources on the Explore tool Dashboard.

From the Toolbox, select . The Explore page, or "Dashboard," shows the number of applications, servers, clusters, hosts, and runtimes that Admin Center is managing. The Dashboard also shows the number of resources that are running, stopped, or in an unknown state. If an alert condition is met, the alert is shown on the Dashboard with a link to the appropriate resource page. The resource name, total, and status also have links to the appropriate resource pages.



- View a summary of all applications, servers, clusters, hosts, or runtimes.

From the Dashboard, select a link on the **Applications**, **Servers**, **Clusters**, **Hosts**, or **Runtimes** panel. For example, to see a summary of all servers, select the **View All** link on the **Servers** panel. The state of all servers is shown. Whether a server is a collective controller or stand-alone server is also shown.



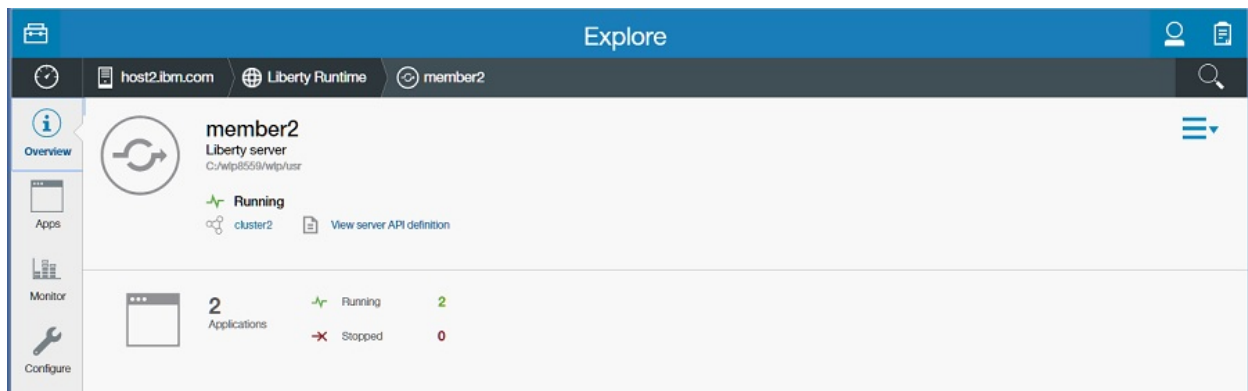
If an alert condition is met, an exclamation mark  is shown on the resource card. Select  for details on the alert condition.

To filter the view, select an icon that shows the number of resources in a specific state. For example, select the **1 Alert** icon to display the one server with an alert condition.


For more timely updates to server and application status, add the websocket-1.1 or websocket-1.0 feature to the server configuration for your servers that use the adminCenter-1.0 feature.

- View details about a resource.

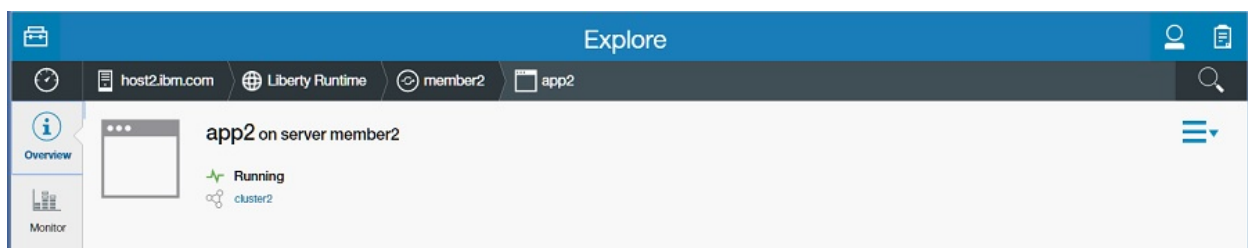
From the Explore page, select a link on the **Applications**, **Servers**, **Clusters**, **Hosts**, or **Runtimes** panel and then a specific resource. For example, to see details about a server, select a link on the **Servers** panel and then one of the servers.



If an alert condition is met, the alert is described with a link to the appropriate resource page.

**8.5.5.9** If the apiDiscovery-1.0 feature is in the configuration file for a server, the  icon with the **View server API definition** link is shown in details about the server. When the server is stopped, the **View server API definition** link is unavailable. For information about the apiDiscovery-1.0 feature, see “Discovering REST API documentation on a Liberty server” on page 1416.

For resources on Liberty servers, select a resource to see more details. For example, select **Apps** and then an application to display details about the application on one of the servers. From the application details page you also can access the Monitor view for each application.



- 8.5.5.4 View details about all servers on a Liberty runtime.

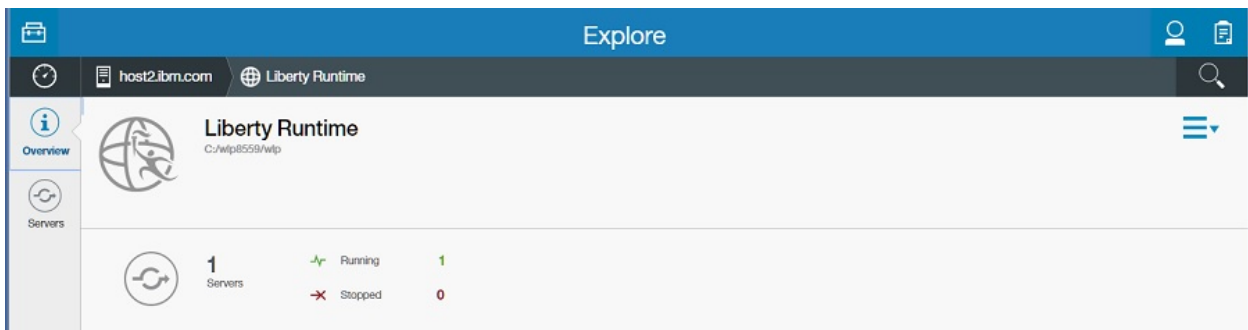
From the Dashboard:

- Select a link on the **Hosts** panel, a host,




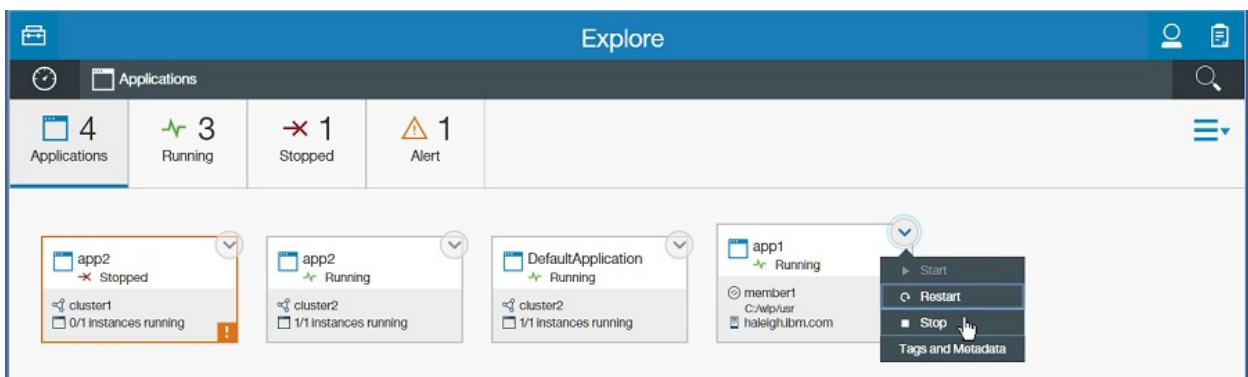
- Select a link on the **Servers** panel, a server, and then **Liberty Runtime** on the horizontal navigation bar.


The Runtime view displays details about the runtime, such as the number and states of servers on the runtime. The runtime details are useful for distinguishing among servers with the same name on the same host.




- Start, stop, or restart a resource.

To start, stop, or restart one resource, select  for the resource, select the available action from the popup dialog, and then confirm the selection.



Alternatively, select  > *one\_or\_more\_resources* > *action\_button*. You can start, stop, or restart multiple resources with this option. For example, to start two stopped servers:

1. When viewing Servers, select .
2. Select two servers in a stopped, partially started, or unknown state.

The screenshot shows the 'Explore' interface with a search bar at the top. A search filter dropdown is open, showing options like Name, Type, State, and Tag. The main table lists applications with columns for Alert, Name, Location, State, Instances, Tags, and Actions. Below it, a section for '4 Servers' lists server details like Name, User Directory, State, Apps, Host, and Tags. A final section for '2 Clusters' is partially visible.

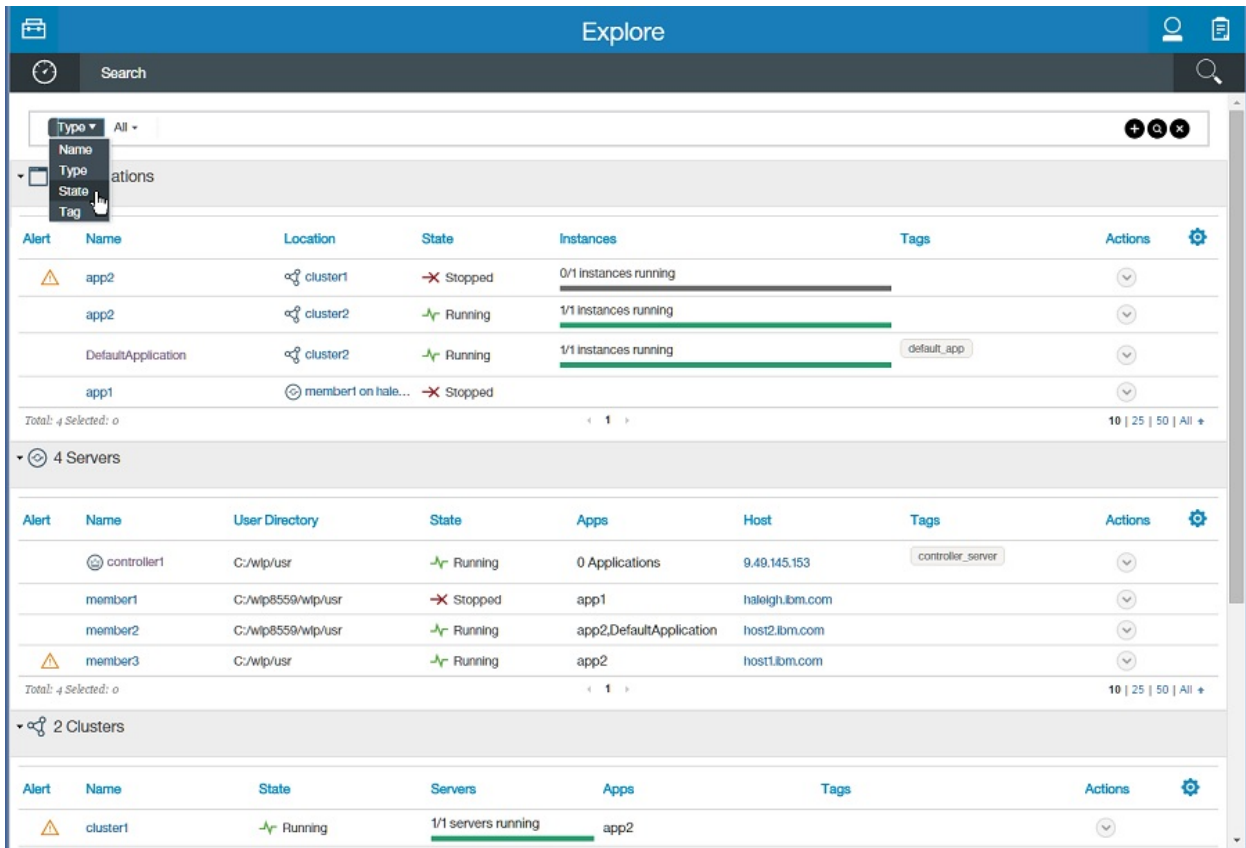
Alert	Name	Location	State	Instances	Tags	Actions
	app2	cluster1	Stopped	0/1 Instances running		
	app2	cluster2	Running	1/1 Instances running		
	DefaultApplication	cluster2	Running	1/1 Instances running	default_app	
	app1	member1 on hale...	Stopped			


Alert	Name	User Directory	State	Apps	Host	Tags	Actions
	controller1	C:\wlp\usr	Running	0 Applications	9.49.145.153	controller_server	
	member1	C:\wlp8559\wlp\usr	Stopped	app1	haleigh.ibm.com		
	member2	C:\wlp8559\wlp\usr	Running	app2,DefaultApplication	host2.ibm.com		
	member3	C:\wlp\usr	Running	app2	host1.ibm.com		

, specify the search criteria, and then select . To add search criteria, select and then specify additional values. To clear the entire search field, select .


The search results list details about resources that fit the search criteria. You can select and then select an available action from the popup dialog to change the resource status or, for hosts, to deploy a server package. To see more details about a resource, select the resource name.



**8.5.5.6** You can filter the search results by entering a string in the search field. For example, enter `er` to show resources that contain the "er" string.

**8.5.5.9** To show or hide columns, select  and the columns to display.

## What to do next

Select  to return to the Explore tool Dashboard. Select a previously viewed resource page on an Explore tool navigation bar to return to that page.

**8.5.5.4** To directly launch the Explore tool in the future, you can use the URL that is shown in the browser when viewing the Explore tool. For example, to directly view information about a particular resource in the future, you can use the URL that is shown in the browser when viewing information about the resource in the Explore tool. Each resource in the Explore tool has a unique URL that you can bookmark and use to directly launch the same resource page.



To return to the Toolbox, select .

## Monitoring metrics in Admin Center

**8.5.5.5**

You can use the Monitor view of the Admin Center Explore tool to track used heap memory, loaded classes, active Java virtual machine (JVM) threads, central processing unit (CPU) usage, and other metrics depending on the resource. The Monitor view shows the metrics graphically in charts. You can customize the Monitor view by selecting the charts to show or hide.

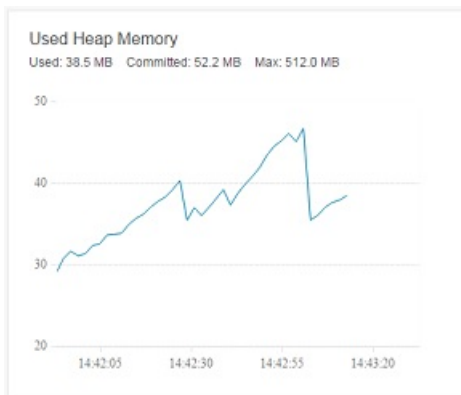
## Procedure

- Open the Monitor view on a server or application.
  1. From the Toolbox, select .
  2. Select a server or application to monitor.
    - To monitor a server, select the **Servers** panel and then a server.
    - To monitor an application, select the **Applications** panel and then an application instance. Or, to monitor an application on a server, select the **Servers** panel, a server, and then an application.
  3. Select  **Monitor** on the vertical navigation bar.

When first displayed, the Monitor view shows the following charts:

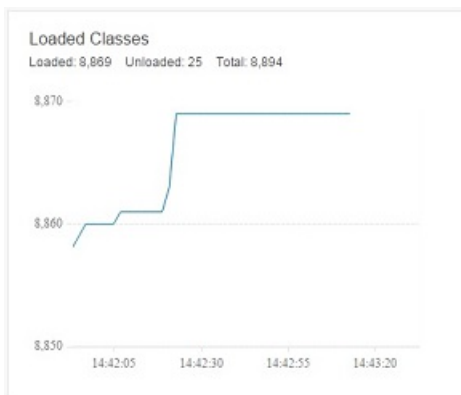
### Used Heap Memory

The Used Heap Memory chart shows the heap memory used by the server in megabytes (MB) every two seconds. The chart also shows the used, committed and maximum megabytes.



### Loaded Classes

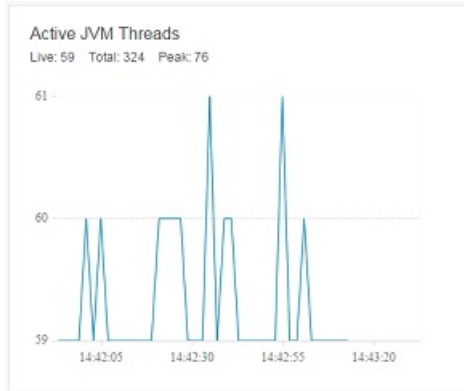
The Loaded Classes chart shows the number of classes loaded every two seconds. The chart also shows the number of loaded and unloaded classes, as well as the total number of classes.



### Active JVM Threads

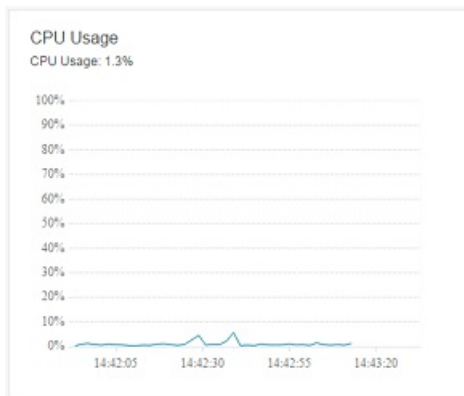
The Active JVM Threads chart shows the number of JVM threads every two seconds. The chart also shows the number of live, total, and peak threads.





### CPU Usage

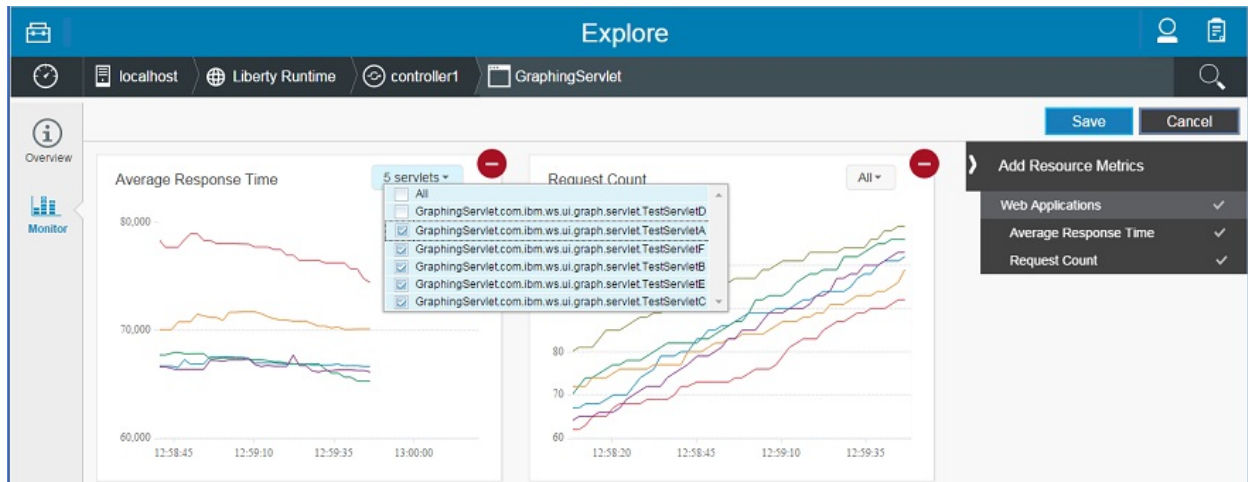
The CPU Usage chart shows the percentage of CPU used every two seconds.




If a server has the `monitor-1.0` feature enabled, the Monitor view also has other charts, depending on the resource:


- Active Sessions
- Active Liberty Threads
- Average Response Time
- Average Wait Time
- Request Count
- Used Connections

**8.5.5.6** The charts available if a server enables the `monitor-1.0` feature have additional configuration options. For example, charts for web applications with multiple servlets, servers with active sessions, or servers with data sources display a drop-down list from which you can select resources to show in the chart.



Select , the resources to show in the chart, and then **Save**.

- Show or hide charts.


Select , the charts to show or hide, and then **Save**. The Monitor view shows or hides the selected charts, depending on whether the chart is shown or hidden when you make the selection.

**8.5.5.6** Your selections are saved for the resource and login user name. When the user opens the Monitor view on the resource in the future, the same chart selections are shown.


- Show or hide chart legends.

If a chart has multiple grid lines, you can choose whether to show or hide the chart legend depending on whether a legend is shown or hidden when you make a selection.

For example, to view a chart legend, select  for the chart and then **Show legend**.

To then hide the legend, select  for the chart and then **Hide legend**.

- **8.5.5.6** View chart data.

In addition to viewing chart data graphically, you can view chart data in a table. Select  for the chart and then **View chart data**.

## Configuration updates

You can make updates to the configuration by using the developer tools or from the command line.

### Updating the server.xml file

The server.xml file can be updated either from the developer tools or from the command line. If there are any problems with the configuration, the specific configuration element updates that have problems do not take effect, but other successful updates are implemented.

This behavior can be changed by updating the onError variable from the default value of WARN to FAIL. If the value is set to FAIL, any problem with the configuration update causes the entire update to fail.

**8.5.5.5** Updates occur if you update any files that are included from the server.xml file or configuration in the configDropins directories.

### Restarting the server with an updated configuration

The WebSphere Application Server Liberty runtime environment caches the currently used configuration so that when you restart the server, the server.xml file is not processed unless there are any changes. If the server.xml file is changed, the cached configuration is updated with the new values. If any problems

are found in the new configuration, the cached configuration values remain in use. The console log displays a warning message for each configuration element that is still using an older value. For example: CWWKG0076W: The previous configuration for httpEndpoint with id defaultHttpEndpoint is still in use.

The server can be started without using cached configuration by running the server script with the `--clean` option.

---

## Liberty and Chef

Chef software is an open source configuration management tool that you can use to create and manage the installation of an Infrastructure as a Service (IaaS). You can use Chef to provision a Liberty installation.

Chef uses cookbooks, which are reusable sets of components that are written in the Ruby programming language. A cookbook provides all the necessary components that are needed to configure an associated piece of software, for example, Apache HTTP Server. An important distinction between using Chef and writing scripts is the ability of Chef to determine the differences between the current software configuration and a new configuration and make only the changes necessary to move from one to the other. For more information, see the online Chef documentation [About Cookbooks](#).

By using Chef, you can create scalable infrastructure with minimal configuration to maintain. You can easily expand your existing infrastructure, for example, by creating and starting a new application server or web server. Chef automatically connects the new servers into the existing infrastructure as necessary.

The `wlp` cookbook installs and configures the WebSphere Application Server Liberty. It provides recipes, resources, and libraries for creating, managing, and configuring Liberty server instances. For more information, see `wlp` cookbook.

To learn more about Chef, see the online Chef documentation [How Chef works](#).

To learn more about using Chef cookbooks, see [Getting started with the Chef cookbooks for Liberty](#).

---

## Including configuration information from external xml files in the server.xml file

8.5.5.6

You can use the **include** element to include configuration information from an external xml file in the `server.xml` file.

If you have configuration information in an external xml file, you can use the **include** element to include the configuration information in the `server.xml` file. For example, if you have an xml file, `simpleSecurity.xml`, with the following content:

```
<server>
 <quickStartSecurity userPassword="thePassword"/>
</server>
```

You can use the following method to include the configuration information in `simpleSecurity.xml` file in your `server.xml` file:

```
<server>
 <featureManager>
 <feature>servlet-3.0</feature>
 </featureManager>
 <quickStartSecurity userName="theUser"/>
 <include location="simpleSecurity.xml"/>
</server>
```

The effective configuration is as follows:

```
<server>
 <featureManager>
 <feature>servlet-3.0</feature>
 </featureManager>
 <quickStartSecurity userName="theUser"/>
 <quickStartSecurity userPassword="thePassword"/>
</server>
```

## Conflict handling

You can configure the *onConflict* attribute in the `server.xml` file to handle the value conflict between `server.xml` file and the external file. This attribute can be configured to one of the three values: *Merge*, *Replace*, and *Ignore*.

**Merge** The values are merged together. *Merge* is the default value of the *onConflict* attribute and *Merge* is equivalent to the behavior that you get if you specify all of the conflicting elements in the `server.xml` file. In the previous example, there are two **quickStartSecurity** elements, and they are effectively merged into a single element. The effective configuration is as follows:

```
<quickStartSecurity userName="theUser" userPassword="thePassword"/>
```

For more information about how configuration elements are merged, see “Configuration element merging rules.”

### *Replace*

The value from the included configuration file replaces the conflicting values in the `server.xml` file. In the previous example, the included **quickStartSecurity** element replaces the one from the `server.xml` file, so the effective configuration is as follows:

```
<quickStartSecurity userPassword="thePassword"/>
```

**Ignore** The value from the included file is ignored. In the previous example, the **quickStartSecurity** element from the included file is ignored, so the effective configuration is as follows:

```
<quickStartSecurity userName="theUser"/>
```

---

## Configuration element merging rules

8.5.5.6

If a configuration element is specified multiple times in the server configuration, the elements are merged. The following rules apply to configuration merging:

- Singleton elements are always merged. In the following example, all instances of the element in the server configuration are `featureManager` merged to form a single `featureManager` element:

```
<featureManager>
 <feature>servlet-3.0</feature>
</featureManager>
<featureManager>
 <feature>jdbc-4.0</feature>
</featureManager>
```

The effective configuration becomes:

```
<featureManager>
 <feature>servlet-3.0</feature>
 <feature>jdbc-4.0</feature>
</featureManager>
```

- Factory elements that are specified at the top level of the server configuration are merged if they have the same ID. In the following example, the `dataSource` element with `id= "ds1"` is merged and the `dataSource` element with `id= "ds2"` stays as it is. For example,

```

<dataSource id="ds1" jdbcDriverRef="myDriver"/>
<dataSource id="ds1" jndiName="jdbc/myDriver"/>
<dataSource id="ds2" jdbcDriverRef="myDriver2"/>

```

The effective configuration becomes:

```

<dataSource id="ds1" jdbcDriverRef="myDriver" jndiName="jdbc/myDriver"/>
<dataSource id="ds2" jdbcDriverDref="myDriver2"/>

```

- If a factory element does not have an ID value, it is considered distinct from other elements of the same type without an ID value. Multiple factory elements without ID values are not merged together. In the following example, the dataSource are not merged, so the effective configuration is the same as the specified configuration:

```

<dataSource jdbcDriverRef="myDriver"/>
<dataSource jndiName="jdbc/myDriver"/>

```

The effective configuration becomes:

```

<dataSource jdbcDriverRef="myDriver"/>
<dataSource jndiName="jdbc/myDriver"/>

```

- If the elements that are to be merged have conflicting attributes, the merged element uses the last value that is encountered by the configuration parser. In the following example, the dataSource element with id= "ds1" is merged and the jdbcDriverRef="myDriver2" is used, while jdbcDriverRef="myDriver" is removed.

```

<dataSource id="ds1" jdbcDriverRef="myDriver"/>
<dataSource id="ds1" jdbcDriverRef="myDriver2"/>

```

The effective configuration becomes:

```

<dataSource id="ds1" jdbcDriverRef="myDriver2"/>

```

- If a factory element is nested underneath another element, it is merged with other elements under the same effective parent only. In the following example, the dataSource element with id= "ds1" is merged and the properties.derby.embedded id="props1" element is merged with the other properties.derby.embedded id="props1" element whose parent is alsodataSource id="ds1".

```

<dataSource id="ds1">
 <properties.derby.embedded id="props1" databaseName="myDB"/>
</dataSource>
<dataSource id="ds2">
 <properties.derby.embedded id="props1" user="myUser"/>
</dataSource>
<dataSource id="ds1">
 <properties.derby.embedded id="props1" createDatabase="create"/>
</dataSource>

```

The effective configuration becomes:

```

<dataSource id="ds1">
 <properties.derby.embedded id="props1" databaseName="myDB" createDatabase="create"/>
</dataSource>
<dataSource id="ds2">
 <properties.derby.embedded id="props1" user="myUser"/>
</dataSource>

```

- If a factory element is nested underneath another element and the factory element does not have a specified ID value, special rules apply depending on the cardinality of the nested element. If multiple nested elements of a particular type are expected, the elements are not merged. However, if only a single nested element is expected, the nested elements are merged together. For example,

```

<topLevel>
 <multipleNested enabled="true"/>
 <multipleNested value="1"/>
 <singleNested enabled="false"/>
 <singleNested value="2"/>
</topLevel>

```

The effective configuration becomes:

```
<topLevel>
 <multipleNested enabled="true"/>
 <multipleNested value="1"/>
 <singleNested enabled="false" value="2"/>
</topLevel>
```

- If two factory elements are nested underneath another element and the ID values that do not match, the merging behavior depends on the cardinality of the nested element. If multiple nested elements are expected, the elements are not merged. If a single nested element is expected, the nested elements are merged together despite the conflicting ID values. For example,

```
<topLevel>
 <multipleNested id="1" enabled="true"/>
 <multipleNested id="2" value="1"/>
 <singleNested id="3" enabled="false"/>
 <singleNested id="4" value="2"/>
</topLevel>
```

The effective configuration becomes:

```
<topLevel>
 <multipleNested id="1" enabled="true"/>
 <multipleNested id="2" value="1"/>
 <singleNested id="4" enabled="false" value="2"/>
</topLevel>
```

---

## Chapter 6. Extending Liberty

You can expand the capability of Liberty by using product extensions. You can write your own Liberty features and install them onto an existing Liberty server, or you can package them for delivery to your users.

### About this task

This section describes how to develop features for a product extension, how to install features to the built-in “usr” product extension, and how to use your features in an application server. Liberty provides various System Programming Interfaces (SPIs) that you can use to extend the runtime environment; you can also use more advanced features such as operating the Liberty server from your Java applications programmatically. The Java API documentation for each Liberty SPI is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

For an overview of writing product extensions for Liberty, see “Product extension” on page 577.

For full details of how to extend Liberty, see the following subtopics:

---

### Developing a Liberty feature for Liberty

A Liberty feature consists of a feature manifest file, and one or more OSGi bundles. The OSGi bundles contain classes and services that provide a particular capability when the feature is installed onto a Liberty server.

#### About this task

You can develop a Liberty feature in either of the following ways:

- Develop the feature manually; see “Developing a Liberty feature manually.”
- Use the WebSphere Application Server Developer Tools; see “Creating a Liberty feature by using developer tools” on page 1106.

For full details on developing Liberty features, see the following subtopics:

### Developing a Liberty feature manually

You can create a Liberty feature manually and install it to Liberty.

#### About this task

A feature can consist of a single OSGi bundle and a feature manifest file. This example makes a library available to applications so that the external packages are visible on the default application class path. By copying the feature manifest into the `${wlp.user.dir}/extension/lib/features` directory, and the OSGi bundle into the `${wlp.user.dir}/extension/lib` directory, the feature can be installed to Liberty. Then you can use the feature in your `server.xml` file.

For details about the format of a feature manifest file, see “Liberty feature manifest files” on page 1097.

This example describes how to construct a Liberty feature manually. Alternatively, you can use the WebSphere Application Server Developer Tools. See “Creating a Liberty feature by using developer tools” on page 1106.

## Procedure

To create a Liberty feature manually, complete the following steps:

1. Create an OSGi bundle containing your Java classes, and a bundle manifest file with appropriate OSGi headers, for example to export the Java packages that you want to expose to applications. `Bundle-SymbolicName` is the only required header; this entry specifies a unique identifier for a bundle, based on the reverse domain name convention. It is good practice to specify a version for the bundle, and in this example some Java packages are exported for application use:

```
Bundle-SymbolicName: com.usr.samplebundle
Bundle-Version: 1.0.1
Export-Package: com.usr.samplebundle.pkg1; version="1.0.0",
 com.usr.samplebundle.pkg2; version="1.0.1"
```

2. Use the `jar` command to package the Java classes and the feature manifest file. For example:  
`jar cfm samplebundle.jar MANIFEST.Mf *.class`
3. Create a feature manifest file named `feature-name.mf` which describes the feature to the runtime environment.
  - a. Provide the required manifest headers:
    - `Subsystem-SymbolicName` to specify the identity and visibility of the feature;
    - `Subsystem-Content` to locate the files that comprise the feature;
    - `IBM-Feature-Version` to identify which version of feature support is required by the runtime environment.
  - b. Best practice: Add the optional manifest headers to indicate the applicable version of the subsystem specification (`Subsystem-ManifestVersion`), the version of your feature (`Subsystem-Version`), and a short name of your feature (`IBM-ShortName`). Specifying these values will help you to evolve your feature in the future.
  - c. In the `IBM-API-Package` header, list the packages that are to be exposed on the default class loader for applications.
  - d. Optional: When you create your Liberty feature, you install it into the user product extension, and the packages in your feature can be accessed by any other feature that is installed to the user product extension. To make one or more SPI packages available to features in other product extensions, list the packages in the `IBM-SPI-Package` header.

```
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: com.example.myfeature.sample-1.0; visibility:=public
Subsystem-Version: 1.0.0.qualifier
Subsystem-Type: osgi.subsystem.feature
Subsystem-Content: samplebundle; version="[1,1.0.100]"
IBM-Feature-Version: 2
IBM-API-Package: com.usr.samplebundle.pkg1; type="api",
 com.usr.samplebundle.pkg2; type="api"
IBM-SPI-Package: com.sample.myservice.spi;
IBM-ShortName: sample-1.0
```

4. Copy the bundle into the `${wlp.user.dir}/extension/lib` directory.
5. Copy the feature manifest into the `${wlp.user.dir}/extension/lib/features` directory.
6. If you have defined `Subsystem-Name` and `Subsystem-Description` headers in the feature manifest file, and have localized the values, copy the localization files specified in the `Subsystem-Localization` header into the `${wlp.user.dir}/extension/lib/features/l10n` directory.

## Results

After your feature is installed to Liberty, you can add the feature name to the list of configured feature in your `server.xml` file. For example:

```
<featureManager>
 <feature>usr:sample-1.0</feature>
</featureManager>
```



## Liberty feature manifest files

A Liberty feature consists of a feature manifest file and a collection of one or more OSGi bundles that provide classes and services corresponding to a particular capability in the Liberty profile runtime environment. You can find the introduction of the format of a feature manifest and the meaning of each header in the manifest file.

The feature manifest file in the Liberty profile uses the Subsystem Service metadata format in the OSGi Enterprise R5 specification. A feature is defined by a feature manifest file (.mf file) that is stored in the lib/features directory, and must use a custom type of Subsystem: `osgi.subsystem.feature`. For more information on OSGi manifest syntax, see section 1.3.2 of the OSGi core specification.

**Note:** In the feature manifest file, the attributes take the form *name=value*, but directives take the form *name:=value*.

The following headers are defined:

*Table 82. Headers of a feature manifest file.*

This table shows the headers of a feature manifest file in the Liberty profile. The first column shows a list of headers. The second column shows the description of each header, and the third column states whether the header is required.

Header	Description	Required?
Subsystem-ManifestVersion	The version format for the feature manifest file. Must be set to 1.	No
Subsystem-SymbolicName	The symbolic name of the feature and any attributes or directives.	Yes
Subsystem-Version	The version of the feature. See the OSGi core specification section 3.2.5 for the details of how this is defined.	No
Subsystem-Type	The subsystem type for the feature. All features are currently of the same subsystem type: <code>osgi.subsystem.feature</code> .	Yes
Subsystem-Content	The subsystem content of the feature. This is a comma separated list of bundles and subsystems that are required to run this feature. If you want to allow the auto feature to be configured in the <code>server.xml</code> file, you must have the capability header containing the required features, and also define them in the subsystem content.	Yes
Subsystem-Localization	The location of the localization files for the feature.	No
Subsystem-Name	A short, human readable name for the feature. This value can be localized.	No
Subsystem-Description	A description of the feature. This value can be localized.	No
IBM-Feature-Version	The version of this subsystem type. Must be set to 2.	Yes
IBM-Provision-Capability	The capability header that describes whether a feature can be provisioned automatically.	No
IBM-API-Package	The API packages that are exposed to applications by this feature, features in other product extensions, and the Liberty kernel.	No
IBM-API-Service	The OSGi services that are exposed to OSGi applications by this feature.	No

Table 82. Headers of a feature manifest file (continued).

This table shows the headers of a feature manifest file in the Liberty profile. The first column shows a list of headers. The second column shows the description of each header, and the third column states whether the header is required.

Header	Description	Required?
IBM-SPI-Package	The SPI packages that are exposed by this feature to features in other product extensions, and the Liberty kernel.	No
IBM-ShortName	The short name of the feature.	No
IBM-AppLiesTo	The Liberty version that this feature applies to.	No
Subsystem-License	The license type for this feature.	No
IBM-License-Agreement	The prefix of the location of the license agreement files.	No
IBM-License-Information	The prefix of the location of the license information files.	No
IBM-App-ForceRestart	Specifies that applications are to be restarted when the feature is installed to, or uninstalled from, a running server.	No

## Subsystem-SymbolicName

The syntax for this header matches the `Bundle-SymbolicName` syntax for a bundle. It has a symbolic name that follows the package names style syntax, and can optionally take a set of attributes and directives.

The following attributes are supported:

- **superseded**. This attribute indicates whether this feature is superseded by one or more features or items of functionality. It takes one of the following values:
  - `true` - The feature is superseded.
  - `false` - The feature is not superseded.

This attribute is optional; the default value is `false`.

For more information, see [Superseded features](#).

- **superseded-by**. This attribute specifies a comma-separated list of the features that supersede this feature, if any, and is optional.

The following directive is supported:

- **visibility**. This directive takes one of the following values:
  - `public` - Feature considered to be API. The feature is supported by the developer tools, for use in the `server.xml` file, and output in messages.
  - `protected` - Feature considered to be SPI. The feature is not supported by the developer tools, for use in the `server.xml` file, or output in messages. The feature is provided so extenders can use it to build higher-level features.
  - `private` - (default) The feature is product internals. The feature is not supported for use in the `server.xml` file or to be referenced by extender features. The feature can be changed at any time, including between fix packs.

For example:

```
Subsystem-SymbolicName: com.ibm.example.feature-1.0;
 visibility:=public; superseded=true; superseded-by="com.ibm.example.feature-2.0"
```

If a feature name in the superseded-by list is surrounded by brackets, [], this feature is separated from the superseding feature. In the following example, feature `appSecurity-1.0`, which contains features `servlet-3.0` and `ldapRegistry-3.0`, is superseded by feature `appSecurity-2.0`, which does not contain `servlet-3.0` and `ldapRegistry-3.0` features:

```
IBM-ShortName: appSecurity-1.0
Subsystem-SymbolicName: com.ibm.websphere.appserver.appSecurity-1.0; visibility:=public;
superseded=true; superseded-by="appSecurity-2.0, [servlet-3.0], [ldapRegistry-3.0]"
```

For more information, see Separated features.

**Best practice:** If the developer tools must show the feature, it must be `public`. If the feature is available only to trusted parties, it must be protected. If the feature is internal and subject to change at any time, it must be `private`.

- **8.5.5.6 singleton.** This directive takes one of the following values:
  - *True*. The feature is a singleton.
  - *False*. The feature is not a singleton.

The **singleton** directive is optional. The default value is *False*.

This directive is used to declare that a particular feature is a singleton. A singleton means that only one version of a given feature can be loaded into the runtime at a time. By default, features are not singletons. If the feature is a singleton and multiple versions of a given feature are required by the server configuration, then the runtime attempts to find a common version that is tolerated by all requiring features. For more information on version toleration, see the **ibm.tolerates** directive under Subsystem-Content.

When a feature is a singleton, then the symbolic name value is in the form "`<singleton feature name>-<singleton version>`", where the name and version are separated by a hyphen. The singleton feature name can contain hyphens, but the characters that follow the last hyphen are interpreted as the singleton version. If the characters that follow the last hyphen are not a valid version, then a singleton version of `0.0.0` is used and the complete symbolic name is used as the singleton name. The singleton version is used when processing `ibm.tolerates` directives under Subsystem-Content; for example:

```
Subsystem-SymbolicName: com.ibm.example.feature-1.0;
visibility:=public; singleton:=true
```

## Subsystem-Content

This header defines the content of the feature, both for run time and install. It follows the same header syntax as the Subsystem specification with the following syntax:

```
Subsystem-Content ::= content (',' content)*
content ::= unique-name (';' parameter)*
unique-name ::= unique-name (see OSGi core spec section 1.3.2)
```

The unique-name uses the form of the `Bundle-SymbolicName` or `Subsystem-SymbolicName` headers. The following attributes are supported:

- **version** - The range of versions to be matched when you find a bundle. Only bundles in this range are selected. A typical example of the version range is `[1,1.0.100)`.
- **type** - The type of content to be provisioned. You can specify any value to indicate the content type; some types result in bundles being installed and started in the OSGi framework of a server that uses the feature, all types cause the content to be included in an installation package that includes the feature. The following values are predefined:
  - `osgi.bundle` - This is the default value and indicates an OSGi bundle that should be provisioned both into the OSGi framework of the server and an installation package.
  - `osgi.subsystem.feature` - This value indicates that the feature should be provisioned both into the OSGi framework of the server and an installation package. These features need to use the name that is specified in `Subsystem-SymbolicName` header.

- `jar` - This value indicates that a JAR file should be included in an installation package and is selected by using a combination of a version range, a location value, or both.
- `file` - This value indicates that the file that is identified in the **location** attribute should be included in an installation package.

The following directives are supported:

- **location** - The location of the bundle. For a bundle or JAR type, this value can be a comma-separated list of directories that represent a search path. For any type, this value might be a single entry that points directly to the resource and might be specified as a file URL. Paths might be absolute or relative. Relative paths are resolved relative to the location of the product extension that contains the feature. For a user feature, the default product extension location is used which is `${wlp.user.dir}/extension`. The location of non-default product extensions is declared by the `com.ibm.websphere.productInstall` property in its properties file in the `${wlp.install.dir}/etc/extensions` directory.

For example:

```
Subsystem-Content: com.ibm.websphere.appserver.api.basics; version="[1,1.0.100)"; type=jar; location="dev/api/ibm/,lib/",
 com.ibm.websphere.appserver.spi.application;
 location="dev/spi/ibm/com.ibm.websphere.appserver.spi.application_1.0.0.jar"; type="jar",
 com.ibm.websphere.appserver.spi.application_1.0.0-javadoc.zip;
 location="dev/spi/ibm/javadoc/com.ibm.websphere.appserver.spi.application_1.0.0-javadoc.zip"; type="file"
```

- **start-phase** - The start phase when the bundle should start during system startup. The **start-phase** directive can take one of the following values:
  - `SERVICE` - This value indicates the earliest phase. By default it maps to a start level of 9.
  - `CONTAINER` - This is the default value if no **start-phase** is provided. It indicates the container phase when application containers are started. By default it maps to a start level of 12.
  - `APPLICATION` - This value indicates the latest phase when applications are started.

Bundles can also be defined to start before or just after these phases by adding `_LATE` to be later, or `_EARLY` to be earlier than the key phase. So if you want to run immediately after the container phase, use `CONTAINER_LATE`, and if you want to run before the `APPLICATION` phase then use `APPLICATION_EARLY`.

- **8.5.5.6 ibm.tolerates** - Specifies alternative singleton version or versions of a singleton feature, `type=osgi.subsystem.feature`, to be provisioned into the system if there are version conflicts.

The unique-name specifies the symbolic name of the preferred version of a singleton feature. If the including feature is known to work with other singleton versions of a given singleton feature, then these singleton versions can be specified by using the **ibm.tolerates** directive. This gives greater compatibility to the defining feature in the case that other features define conflicting required version values of a given singleton feature.

Singleton versions that are listed in the **ibm.tolerates** directive are only used if a version conflict. The ordering of versions that are listed in the **ibm.tolerates** directive is not significant - any version that is listed in the **ibm.tolerates** directive can be selected to satisfy dependency requirements.

The tolerated version or versions of a given singleton feature must be explicitly listed in the **ibm.tolerates** directive. Use commas to separate a list of tolerated versions. Specifying a version range is not supported.

For example:

```
Subsystem-Content: com.ibm.websphere.appserver.example.featureA-1.1; ibm.tolerates="1.2"; type="osgi.subsystem.feature",
 com.ibm.websphere.appserver.example.featureB-1.1; ibm.tolerates="1.2, 1.4, 1.6"; type="osgi.subsystem.feature"
```

#### Note:

Tolerated versions are not transitive. This prevents a feature that your feature depends on from being automatically opted in to supporting a later level of a feature, without testing it.

For example: User feature `featureC-1.1` includes `sipServlet-1.1` in the `Subsystem-Content` header of its manifest file. `sipServlet-1.1` includes `servlet-3.0` and tolerates `servlet 3.1`. If `featureC-1.1` was

written before servlet-3.1 existed and then servlet-3.1 was added and tolerated by the feature used by it (sipServlet-1.1), featureC-1.1 should have a say on whether it also tolerated servlet-3.1.

If you configure the server.xml file to have the following two features:

```
<feature>usr:featureC-1.1</feature> // includes: sipServlet-1.1
<feature>websocket-1.0</feature> // this feature requires servlet-3.1
```

You will see an error message that resembles the following displayed:

```
CWWKF0033E: The singleton features servlet-3.0 and servlet-3.1 cannot be loaded at the same time.
The configured features usr:featureC-1.1 and websocket-1.0 include one or more features that cause the conflict.
Your configuration is not supported; update server.xml to remove incompatible features."
```

This error is reported because featureC-1.1 is not opted in to tolerating servlet-3.1, and so has to have servlet-3.0, and websocket-1.0 does not support servlet-3.0 and so has to have servlet-3.1.

The solution is for featureC-1.1 to also directly depend on servlet-3.0 and tolerate servlet-3.1.

 The following directive is supported in version 8.5.5.4 and later:

- **ibm.executable** - Adds the execute permission to the associated file, according to the current umask setting, when the value is set to "true". Any other value results in no action taken. The following table shows the current umask and which class gets the execute permission.

Table 83. Examples of umask values and classes with execute permissions set by *ibm.executable*

Umask	Execute permissions granted to class
022	owner, group, other
023	owner, group
055	owner

## Subsystem-Localization

This header specifies the location of the localization files for the feature.

For example:

```
Subsystem-Localization: OSGI-INF/110n/loc
```

## Subsystem-Name

Use this header to supply a short, human readable name for the feature. You can specify either a literal string or a property name. If you specify a property name, the value can be localized.

For example

```
Subsystem-Name: %name
```

where the value of name is defined in a properties file at the location that is specified by the Subsystem-Localization header (loc.properties in the previous example), in the following format:

```
name=feature_name
```

## Subsystem-Description

Use this header to supply a description for the feature. You can specify either a literal string or a property name. If you specify a property name, the value can be localized.

For example

```
Subsystem-Description: %desc
```

where the value of desc is defined in a properties file at the location that is specified by the Subsystem-Localization header (loc.properties in the previous example), in the following format:

```
desc=feature_description
```

## IBM-Provision-Capability

Automatically provisioned features are features that have the IBM-Provision-Capability header in the manifest. This header describes other features that must be provisioned for this feature to be automatically provisioned. When you list the other features, use the Subsystem-SymbolicName header of the feature. When any features are configured in the server.xml file, the runtime checks to see whether any automatically provisioned features have their capabilities satisfied, and if any have, they are automatically provisioned.

The format of the IBM-Provision-Capability header uses standard OSGi LDAP filters.

## IBM-API-Package

This header is used to indicate which API packages are visible to applications. It matches the Export-Package header syntax. This means it is a comma-separated list of API packages, but each API package can have some attributes.

The following attribute is supported:

- **type** - The type of API package. The **type** attribute takes one of the following values:
  - spec - Indicates an API provided by a standard body, such as javax.servlet or org.osgi.framework.
  - ibm-api - Indicates a value-add API provided by IBM.
  - api - Indicates a user-defined API. This is the default value.
  - third-party - Indicates an API that is visible, but not controlled by IBM. Typically, these are open source packages.
  - internal - Indicates non-API packages that must be exposed to applications for them to function. This might be used if Java code is bytecode enhanced, or *waved*, to add references to internal code at run time.

For example:

```
IBM-API-Package: javax.servlet; type="spec",
 com.ibm.websphere.servlet.session; type="ibm-api",
 com.ibm.wsspi.webcontainer.annotation; type="internal"
```

## IBM-API-Service

This header is used to indicate which services from the feature are visible to OSGi applications. The feature must also register the service in the OSGi service registry.

It has the following syntax

```
IBM-API-Service ::= service (',' service)*
 service ::= service-name (';' attribute)*
 service-name ::= unique-name
```

The service-name is the Java class or interface name of the service. The attributes are interpreted as the service properties for the services.

For example:

```
IBM-API-Service: com.ibm.example.service.FeatureServiceOne;
 myServiceAttribute=myAttributeValue,
 com.ibm.example.service.FeatureServiceTwo
```

If an OSGi application wants to use the services that are provided by the IBM-API-Service header, the application must include a blueprint reference to the service in order for the service to be provisioned into the application.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
 <reference id="FeatureServiceOneRef"
 interface="com.ibm.example.service.FeatureServiceOne" />
</blueprint>
```

In order for a service to be usable by a bundle in an OSGi application, the interface package must be available to that bundle, which means the interface package must be specified by an Import-Package header in the manifest file of the consuming bundle. The interface package must also be specified by an Export-Package header in a feature bundle and specified in the IBM-API-Package header of the feature manifest file. The service that is provided by a feature must be registered in the OSGi service registry by using the OSGi BundleContext interface or any other mechanism such as Declarative Services or Blueprint. For more information, see “Developing an OSGi bundle with simple activation” on page 1111 and “Composing advanced features by using OSGi Declarative Services” on page 1116.

## IBM-SPI-Package

When you create your own Liberty feature, you install it into the user product extension. All the packages in your feature can be accessed by any other feature that is installed into the user product extension. However, if you want a package in your feature to be accessed by a feature that is installed into another product extension, you must list the package name in the IBM-SPI-Package header.

Any package that is listed in the IBM-SPI-Package header must be exported by a bundle in the Liberty feature, by being listed in the Export-Package header of the bundle manifest file.

## IBM-ShortName

This header is a short name for a feature that you can use to specify a feature in the server.xml file. If there is no IBM-ShortName header in the manifest file, then the Subsystem-SymbolicName is used by default. The IBM-ShortName header is only valid for public features.

## IBM-AppliesTo

This header specifies the Liberty version that this feature applies to. Supply a comma-separated list of items, each in the following form:

```
product_id; productVersion=product_version; productInstallType=product_install_type; productEdition=product_editions
```

If you supply more than one item, the value of *product\_id* must be different for each one.

**8.5.5.7** The value of *productVersion* can be either an exact version, such as 8.5.5.7, or a minimum version, denoted by the version that ends with a plus sign, +, such as 8.5.5.7+.

The value of *productEdition* can be either a single edition or a comma-separated list of editions that are enclosed in quotation marks.

For example:

```
IBM-AppliesTo: com.ibm.websphere.appserver; productVersion=8.5.5.6; productInstallType=Archive; productEdition="BASE,DEVELOPERS,EXPRESS,ND"
```

## Subsystem-License

This header defines the license type for this feature. If you supply a value for the Subsystem-License header, and do not supply values for the IBM-License-Agreement and IBM-License-Information headers, then the Subsystem-License header value is displayed to the user for acceptance during installation.

If there is a feature that is already installed with the same Subsystem-License header value, then the license is not displayed, and license approval is not sought, during the installation. If dependencies in the Subsystem-Content header mean that there are two or more features being installed that have the same Subsystem-License header value, the user has only to accept the license once during installation.

For example:

```
Subsystem-License: L-JTHS-93TMHH
```

```
Subsystem-License: http://www.apache.org/licenses/LICENSE-2.0.html
```

## IBM-License-Agreement

This header specifies the prefix of the location of the license agreement files. Supply the file path in the subsystem archive to the LA\_ *language* files, up to, but not including, the "\_" character (the language code is appended by the installation tool). If this license has not been accepted, the user must accept the license when you install the feature. The license files are copied to the Liberty installation directory.

For example:

```
IBM-License-Agreement: lafiles/LA
```

## IBM-License-Information

This header specifies the prefix of the location of the license information files. Supply the file path in the subsystem archive to the LI\_ *language* files, up to, but not including, the "\_" character (the language code is appended by the installation tool). If this license has not been accepted, the user must accept the license when you install the feature. The license files are copied to the Liberty installation directory.

For example:

```
IBM-License-Information: lafiles/LI
```

## IBM-App-ForceRestart

This header causes applications to be restarted when the feature is installed to, or removed from, a running server. This header can take one of the following values:

- `install` - restart applications when the feature is installed.
- `uninstall` - restart applications when the feature is uninstalled.
- `install,uninstall` - restart applications when the feature is installed or uninstalled.

## Example feature manifest file

The following example shows the definition for the `example-1.0` feature. The `public visibility` attribute allows this feature to be directly specified in server configuration (`server.xml`) files; it will also be included in the drop down list of features that are displayed in **Server Configuration** view of the developer tools and will be available for inclusion in features that are in other product extensions. If this feature is installed into the `usr` product extension of a runtime install, it can be configured into a server by including the following code in the `server.xml` file:

```
<featureManager>
 <feature>usr:example-1.0</feature>
</featureManager>
```



Configuration of this feature in a server results in the specified bundle, `com.ibm.example.bundle1`, being installed and started in the OSGi framework of the server runtime environment. The single API package, `com.ibm.example.publicapi`, will be visible to all applications in that server, except for Java EE applications that are configured to not have visibility to the `api` package type. OSGi applications must explicitly import the package if they wish to use it. The two SPI packages, `com.ibm.example.spi.utils` and `com.acme.spi.spiservices`, will be visible to all feature code in the server, as will the API package.

```
IBM-Feature-Version: 2
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: com.ibm.example-1.0; visibility:=public
Subsystem-Version: 1.0.0.qualifier
Subsystem-Type: osgi.subsystem.feature
Subsystem-Content: com.ibm.example.bundle1; version="1.0.0"
Subsystem-Localization: OSGI-INF/l10n/loc
Manifest-Version: 1.0
Subsystem-Name: %name
Subsystem-Description: %desc
IBM-API-Package: com.ibm.example.publicapi; type="api"
IBM-SPI-Package: com.ibm.example.spi.utils, com.ibm.example.spi.spiservices
IBM-ShortName: example-1.0
```

## Auto-provisioning a feature

Auto-provisioning allows a feature to have dependencies on features that must be provisioned before it can be provisioned.

### About this task

An auto-provisioned feature is a feature that has dependencies on other features. Because of the dependencies, the lifecycle of the auto-provisioned feature is as follows:

- The feature is provisioned automatically when all required features are provisioned.
- The feature is de-provisioned automatically when any of the required features are de-provisioned.

### Procedure

To configure a feature to be auto-provisioned, follow these steps:

1. Determine which features must be provisioned before the runtime automatically provisions this feature.
2. Add the `IBM-Provision-Capability` to the manifest header. The format of the `IBM-Provision-Capability` header uses standard OSGi LDAP filters.
3. Deploy the feature to the server.

### Results

The feature automatically provisions when the required features are provisioned.

### Example

In the following example, if features `requiredFeature1-1.0` and `requiredFeature2-1.0` are provisioned, this feature is automatically provisioned. If either of these required features are removed from the `server.xml` file, this feature is automatically de-provisioned.

```
IBM-Provision-Capability: osgi.identity; filter="(
&(type=osgi.subsystem.feature)(osgi.identity=requiredFeature1-1.0))", osgi.identity;
filter="(&(type=osgi.subsystem.feature)(osgi.identity=requiredFeature2-1.0))"
```

### Automatic installation of auto-provisioned features

If all the required features are also installed, auto-provisioned features can be installed automatically.

To configure a feature to be auto-installed, the `IBM-Install-Policy` header must be added to the feature manifest. The header is optional. If the `IBM-Install-Policy` header is specified, the following values are valid:

- `manual`: The feature is not auto-installed.
- `when-satisfied`: If all required features are installed, the feature is auto-installed.

If the header is not set, then the feature is not auto-installed, which is equivalent to setting the `IBM-Install-Policy` header to `manual`.

## Creating a Liberty feature by using developer tools

You can use the WebSphere Application Server Developer Tools to write your own features and install them into an existing Liberty server, or to package them for delivery to your users.

### About this task

To develop a Liberty feature in the WebSphere Application Server Developer Tools, you create a Liberty feature project and target it to the WebSphere Application Server Liberty version 8.5.5 or later.

You add OSGi bundles that contain classes and services that implement the function provided by your Liberty feature. If your feature provides any API packages to OSGi applications, or SPI packages to features in other product extensions, you can declare those packages in the Liberty feature manifest file.

You can export your liberty feature as a compressed file that can be extracted over an existing WebSphere Application Server Liberty to extend its capabilities.

For more information on creating Liberty features, see “Product extension” on page 577.

Creating a Liberty feature by using WebSphere Application Server Developer Tools is described in more detail in the following subtopics:

### Procedure

1. “Creating a Liberty feature project.”
2. “Adding OSGi bundles to a Liberty feature project” on page 1107.
3. “Specifying API and SPI packages for a Liberty feature project” on page 1107.
4. “Installing a Liberty feature to Liberty V8.5.5” on page 1108.

### Creating a Liberty feature project

To develop a Liberty feature by using the WebSphere Application Server developer tools, you must create a Liberty feature project in your workspace.

### About this task

A Liberty feature project contains classes and services that implement the function provided by your Liberty feature

### Procedure

To create a Liberty feature project, complete the following steps:

1. Click **File > New > Other > OSGi > Liberty Feature Project** and then click **Next**. The New Liberty Feature Project wizard opens.
2. In the **Project name** field, enter the name of your Liberty feature project.

3. Select a **Target runtime** from the drop down list. The list will include the WebSphere Application Server Liberty 8.5.5 if it is defined in your workspace as an installed runtime environment.
4. Click **Next**. The OSGi Bundles Selection dialog box opens.
5. Select one or more OSGi bundles to add to the Liberty feature project, or click **New Bundle** to create an OSGi bundle to add to the Liberty feature project. You can add further bundles after you have created the Liberty feature project; see “Adding OSGi bundles to a Liberty feature project.”  
For information on creating an OSGi bundle, see Creating OSGi bundle projects.
6. Click **Finish** to create the Liberty feature project.

## Results


The Liberty feature project is added to your workspace.

### Adding OSGi bundles to a Liberty feature project

A Liberty feature includes OSGi bundles that contain classes and services. The classes and services implement the functions that the Liberty feature provides. You can include OSGi bundles in a Liberty feature that was created with the WebSphere Application Server Developer Tools by adding the bundles to the corresponding Liberty feature project.

## Procedure

To add OSGi bundles to a Liberty feature project, complete the following steps:

1. From the Project Explorer view, open the feature manifest file for the Liberty feature project by double-clicking the **Manifest** node in the project hierarchy, indicated by the manifest icon (  ).
2. In the Contained Bundles pane, click **Add** to select one or more bundles to add to the Liberty feature project, or click **New** to create a new OSGi bundle to add to the Liberty feature project. For information on creating an OSGi bundle, see Creating OSGi bundle projects.
3. (Optional) Specify the version range for the contained bundle by selecting the bundle, clicking **Properties**, and entering the required values in the **Minimum Version** and **Maximum Version** fields.
4. (Optional) Use the **Location** field in the Properties dialog box to specify the location where you want the bundle to be packaged when exported, relative to the product extension installation folder. If you want the bundle to be packaged in more than one location, enter the locations as a comma-separated list. By default, the bundle is packaged in the `/lib` folder.

## Results

The bundle names are added to the Subsystem-Content header in the manifest file. For more information on the headers in the feature manifest file for a Liberty feature, see “Liberty feature manifest files” on page 1097.

### Specifying API and SPI packages for a Liberty feature project


Use the Liberty feature manifest file to declare which packages you want to share as an API or SPI with other applications and features in the Liberty runtime environment.

## About this task

A package cannot be declared as an API or SPI unless it is exported by a bundle in the Liberty feature, by being listed in the Export-Package header of the bundle manifest file.

## Procedure

To specify API and SPI packages for a Liberty feature project, complete the following steps:

1. From the Project Explorer view, open the feature manifest file for the Liberty feature project by double-clicking the **Manifest** node in the project hierarchy, indicated by the manifest icon (  ).
2. To make one or more API packages available to OSGi applications, click **Add** in the IBM API Packages pane.
3. When you create your own Liberty feature, you install it into the user product extension, and all the packages in your feature can be accessed by any other feature that is installed into the user product extension. To make one or more SPI packages available to features in other product extensions, click **Add** in the IBM SPI Packages pane.
4. (Optional) Specify the package version by selecting the package, clicking **Properties**, and entering the required value in the **Version** field.
5. (Optional) For an API package, select the package type from the **Type** list in the Properties dialog box. The type can be one of the following values:
  - `spec` - Indicates an API provided by a standard body, such as `javax.servlet` or `org.osgi.framework`.
  - `ibm-api` - Indicates a value-add API provided by IBM.
  - `api` - Indicates a user-defined API. This is the default value.
  - `third-party` - Indicates an API that is visible, but not controlled by IBM. Typically, these are open source packages.
  - `internal` - Indicates non-API packages that must be exposed to applications for them to function. This might be used if Java code is bytecode enhanced, or *weaved*, to add references to internal code at run time.

## Results

The package names are added to the IBM-API-Package and IBM-SPI-Package headers in the feature manifest file. For more information on the headers in the feature manifest file for a Liberty feature, see “Liberty feature manifest files” on page 1097.

## Installing a Liberty feature to Liberty V8.5.5

When you develop a Liberty feature by using the WebSphere Application Server Developer Tools, you need to create a Liberty feature project that packages the Liberty feature. You can use the workbench to install Liberty features to Liberty runtime environments and enable installed features using the server configuration editor. Changes to features already installed on Liberty can be pushed to all respective runtime environments using the **Update Feature** menu option in the workbench.

## Before you begin

Create a Liberty feature project.

**Restriction:** This topic is supported for WebSphere Application Server Liberty V8.5.5. For versions 8.5 or earlier, see “Manually installing a Liberty feature to Liberty V8.5 or earlier” on page 1109 topic.

## Procedure

To install a Liberty feature, complete the following steps:

1. In the **Enterprise Explorer** view, right-click your Liberty feature project and select **Install Feature**.
2. In the **Feature install** wizard and under **Target Runtimes**, select the Liberty runtime environment that you want to install your feature. Click **Finish**.

**Tip:** If the feature is already installed on Liberty, the Liberty entry is no longer an available option in the list of target runtimes. Instead, you should use the **Update Features** menu option (available when you right-click the Liberty feature project in the Enterprise Explorer view) to update any changes to a feature already installed on Liberty.

3. Add the feature name to the list of configured features in your server configuration (server.xml file):
  - a. In the Servers view, expand your Liberty server, right-click the **Server Configuration [server.xml]** and select **Open**.
  - b. In the Server Configuration editor, under the **Configuration Structure**, expand **Server Configuration** and select **Feature Manager**.
  - c. Under the **Feature Manager**, select the **Add** button.
  - d. In the **Add Features** wizard, search and select your feature with the prefix **usr:** followed by the name of your Liberty feature project, for example, **usr:MyLibertyFeatureProject**. Click **OK**.

In the **Source** tab of the Server Configuration editor, the server.xml file displays the newly added feature entry under the featureManager node:

```
<featureManager>
 <feature>usr:MyLibertyFeatureProject</feature>
</featureManager>
```

## Results

After installing the Liberty feature, you can find the following file structure in the `${wlp.user.dir}/extension` directory:

```
/lib
 /features
 manifest files
 .
 .
 .
 OSGi bundle JAR files
 .
 .
 .
```

## What to do next

To update changes to a feature already installed on the Liberty runtime environments, use the **Update Features** menu option (available when you right-click the Liberty feature project in the Enterprise Explorer view). A **Progress Information** window opens and the workbench takes a moment to perform this update action.

## Manually installing a Liberty feature to Liberty V8.5 or earlier

When you develop a Liberty feature by using the WebSphere Application Server Developer Tools, you create a Liberty feature project that packages the Liberty feature in a compressed file. To install the Liberty feature, you must extract the contents of the compressed file to the Liberty environment.

## Before you begin

Create a Liberty feature project.

**Restriction:** This topic is supported for WebSphere Application Server Liberty V8.5 or earlier. For versions 8.5.5, see “Installing a Liberty feature to Liberty V8.5.5” on page 1108 topic.

## About this task

The compressed file has the following structure:

```
/lib
 OSGi bundle JAR files
 .
 .
/features
 manifest file
```

## Procedure

To install a Liberty feature, complete the following steps:

1. In your workspace, right-click your Liberty feature project and select **Export > Liberty Feature**.
2. In the **To ESA file** field, specify the location and name of the compressed file to which you want to export the Liberty feature project.
3. Click **Finish** to export the Liberty feature project to the specified location.
4. Extract the contents of the compressed file to the `${wlp.user.dir}/extension` directory.
5. Add the feature name to the list of configured features in your `server.xml` file; you must prefix the feature name with `usr:`.

For example:

```
<featureManager>
 <feature>usr:sample-1.0</feature>
</featureManager>
```

## Adding OSGi metatype descriptions to a Liberty feature project

You can add OSGi metatype descriptions to a Liberty feature that was created with the WebSphere Application Server Developer Tools by creating a metatype XML file. You can package that file in an OSGi bundle project.

## Procedure

1. From the Project Explorer view, add a folder in the `BundleContent` folder, and name it `OSGI-INF`.
2. Create a folder that is named `metatype` in the `OSGI-INF` folder.
3. Create the metatype XML file in the `metatype` folder.

The metatype XML file must have a `.xml` suffix. You can use any name for the file.

4. Optional: Provide translated strings for your metatype definitions.

- a. Create a folder to contain the translated properties files.

For example, you can create a subfolder that is named `I10n` in the `OSGI-INF` folder, and use the prefix `metatype` for your translated properties files.

- b. In the metatype XML file, specify the location of the folder that you created.

Use the following example as a guide:

```
<box>
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
localization="OSGI-INF/I10n/metatype">
</box>
```

## Results

You created a metatype XML file, which you can use to add OSGi metatype descriptions to a Liberty feature.

## What to do next

You can add information to the metatype XML file. For more information about what to add to the file, see “Liberty feature manifest files” on page 1097.

To describe your configuration by using the OSGi metatype service, package the metatype XML file in the OSGI-INF/metatype folder of one of your OSGi bundle projects, not in the Liberty feature project. For best results, put the metatype XML file in the same OSGi bundle as the code that receives and processes the configuration values, such as the associated ManagedService implementation. For more information, see “Describing configuration by using the OSGi Metatype service” on page 1121.

## Developing an OSGi bundle with simple activation

The most straightforward way to control the lifecycle of your OSGi bundle code is to implement the `org.osgi.framework.BundleActivator` interface in one of the classes within your bundle. When the server starts and stops the bundle, the start and stop methods of the BundleActivator interface are called.

### About this task

If you are using the WebSphere Application Server Developer Tools, create an OSGi bundle project, and create an OSGi BundleActivator class in that project. Then identify your bundle activator class to the OSGi framework by adding the Bundle-Activator header to the bundle MANIFEST.MF file. For example: Bundle-Activator: com.example.bundle.Activator.

### Example

```
package com.example.bundle;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

public class Activator implements BundleActivator {
 public void start(BundleContext context) throws Exception {
 System.out.println("Sample bundle starting");
 // Insert bundle activation logic here
 }

 public void stop(BundleContext context) throws Exception {
 System.out.println("Sample bundle stopping");
 // Insert bundle deactivation logic here
 }
}
```

## Receiving configuration data by using the ManagedService interface

Liberty configuration is managed by the OSGi Configuration Admin service and can be accessed according to the OSGi Configuration Admin service specification. Sets of configuration properties are identified by a persisted identity (PID) that is used to associate an element in the server.xml file, where the PID is used as the element name, with a component that registers to receive the properties.

### About this task

For an OSGi bundle whose lifecycle is managed by using the BundleActivator interface, a straightforward way to receive the configuration properties is to implement the `org.osgi.service.cm.ManagedService` interface, which specifies the PID as one its properties.

### Example

#### Remember:

1. In Eclipse, you must select an SPI target runtime from **Window > Preferences > Plug-In Development > Target Platform**.
2. Add the following statement to your MANIFEST.MF file:  
`Import-Package: org.osgi.service.cm;version="1.5.0"`
3. Press Ctrl + Shift + O to update your bundle activator.

In this example, the Activator class implements the ManagedService interface in addition to the BundleActivator interface, and receives configuration properties by using the updated method. You can provide default property values to simplify what must be specified in the user configuration.

```
public class Activator implements BundleActivator, ManagedService {

 public void start(BundleContext context) throws Exception {
 System.out.println("Sample bundle starting");
 // register to receive configuration
 ServiceRegistration<ManagedService> configRef = context.registerService(
 ManagedService.class,
 this,
 getDefaults()
);
 }

 public void stop(BundleContext context) throws Exception {
 System.out.println("Sample bundle stopping");
 configRef.unregister();
 }

 Hashtable getDefaults() {
 Hashtable defaults = new Hashtable();
 defaults.put(org.osgi.framework.Constants.SERVICE_PID, "simpleBundle");
 return defaults;
 }

 public void updated(Dictionary<String, ?> properties) throws ConfigurationException {
 if (properties != null)
 {
 String configColor = (String) properties.get("color");
 String configFlavor = (String) properties.get("flavor");
 }
 }
}
```

User configuration for the bundle can optionally be provided in the server.xml file, or an included file, by the following entry:

```
<simpleBundle color="red" flavor="raspberry" />
```

**Note:** The element name in the user configuration, *simpleBundle* matches the value of the org.osgi.framework.Constants.SERVICE\_PID property used in the ManagedService registration.

For more advanced configuration use, see “Describing configuration by using the OSGi Metatype service” on page 1121.

## Working with the OSGi service registry

You can create an object and register it as an OSGi service for use by third-party features that are deployed to Liberty.

### About this task

Services are the OSGi lightweight and flexible component model. When you create services and wire them together with Java code, you can use mechanisms such as ServiceTrackers to help find the services that you want, and Declarative Services (DS) and Blueprint to specify the wiring declaratively. Liberty has standardized on using DS for wiring, except for a small number of cases where extra flexibility is required.



## Registering OSGi services:

You can create an object and register it as an OSGi service for use by third-party features.

### About this task

By using plain old Java code, you can create an object, and then register it as a service using the `BundleContext` class. Because the code has to run, you typically register the object in a `BundleActivator` interface. When you register the object, you can specify what interfaces it provides, and supply a property map. A `ServiceRegistration` object is returned; if necessary, you can use the `ServiceRegistration` object to change the properties at any time. When the service is completed, you use the `ServiceRegistration` object to unregister the service.

To obtain a service, you query the `BundleContext` for a service that implements a required interface and, optionally, supply an LDAP-syntax filter to match the service properties. Depending on the method you call, you can retrieve the best match or all the matches. You can then use the returned `ServiceReference` that provides the properties to do further matching in your code. You can use the `ServiceReference` to get the actual service object. When you have finished using the service, you use the `BundleContext` to release the service.

### Procedure

1. Declare the service interface by adding the following code in your bundle.

```
package com.ibm.foo.simple;

/**
 * Our multifunctional sample interface
 */
public interface Foo
{
}
```

2. Specify the implementation code of the interface.

```
package com.ibm.foo.simple;

/**
 * The implementation of the Foo interface
 */
public class FooImpl implements Foo
{
 public FooImpl()
 {
 }
 public FooImpl(String vendor)
 {
 }

 /**
 * used by the ServiceFactory implementation.
 */
 public void destroy() {
 }
}
```

3. Use the `BundleContext` to register the service, modify the service properties, and unregister the service directly in your code.

```
import java.util.Dictionary;

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceRegistration;
```

```

/**
 * Registers and unregisters a Foo service directly,
 * and shows how to modify the service properties in code.
 */
public class FooController
{
 private final BundleContext bundleContext;
 private ServiceRegistration<Foo> sr;

 public FooController(BundleContext bundleContext)
 {
 this.bundleContext = bundleContext;
 }

 public void register(Dictionary<String, Object> serviceProperties) {
 Foo foo = new FooImpl();
 //typed service registration with one interface
 sr = bundleContext.registerService(Foo.class, foo, serviceProperties);
 //or
 //untyped service registration with one interface
 sr = (ServiceRegistration<Foo>)bundleContext.registerService(
 Foo.class.getName(), foo, serviceProperties);
 //or
 //untyped service registration with more than one interface (or class)
 sr = (ServiceRegistration<Foo>)bundleContext.registerService(new String[] {
 Foo.class.getName(), FooImpl.class.getName()}, foo, serviceProperties);
 }

 public void modifyFoo(Dictionary<String, Object> serviceProperties) {
 //with the service registration you can modify the service properties at any time
 sr.setProperties(serviceProperties);
 }

 public void unregisterFoo() {
 //when you are done unregister the service using the service registration
 sr.unregister();
 }
}

```

#### 4. Obtain and return the service from another class:

```

package com.ibm.foo.simple;

import java.util.Collection;

import org.osgi.framework.BundleContext;
import org.osgi.framework.InvalidSyntaxException;
import org.osgi.framework.ServiceReference;

/**
 * A simple Foo client that directly obtains the Foo service and returns it when done.
 */
public class FooUser
{
 private final BundleContext bundleContext;

 public FooUser(BundleContext bundleContext)
 {
 this.bundleContext = bundleContext;
 }

 /**
 * assume there's only one Foo

```

```

 */
 public void useFooSimple() {
 ServiceReference<Foo> sr = bundleContext.getServiceReference(Foo.class);
 String[] propertyKeys = sr.getPropertyKeys();
 for (String key: propertyKeys) {
 Object prop = sr.getProperty(key);
 //think about whether this is the Foo we want....
 }
 Foo foo = bundleContext.getService(sr);
 try {
 //use foo
 } finally {
 //we're done
 bundleContext.ungetService(sr);
 }
 }

 /**
 * Use a filter to select a particular Foo. Note we get a collection back and have to pick one.
 * @throws InvalidSyntaxException
 */
 public void useFooFilter() throws InvalidSyntaxException {
 Collection<ServiceReference<Foo>> srs = bundleContext.getServiceReferences(
 Foo.class, "(service.vendor=IBM)(id='myFoo')");
 ServiceReference<Foo> sr = srs.iterator().next();
 String[] propertyKeys = sr.getPropertyKeys();
 for (String key: propertyKeys) {
 Object prop = sr.getProperty(key);
 //think about whether this is the Foo we want....
 }
 Foo foo = bundleContext.getService(sr);
 try {
 //use foo
 } finally {
 //we're done
 bundleContext.ungetService(sr);
 }
 }
}

```

### Using OSGi services:

Services can be registered and unregistered asynchronously at any time. Therefore you should call a service for as short a time as possible. You can use the `ServiceTracker` class to track service availability concurrently.

### About this task

If you want to track services, you can create a `ServiceTracker` object by using your bundle context, the interface you want, and the properties you want to match, and then open the tracker. You can query the tracker for the best match or all matches. Make sure that you do not occupy the service after you use it. You do not have to tell the tracker you are done; the tracker caches the matching services internally, and clears them as they are unregistered. When you have finished using the tracker, use the `serviceTracker.close()` method to close it.

### Example

The following example shows how to use a `ServiceTracker` object to track a service:

```

package com.ibm.foo.tracker;

import com.ibm.foo.simple.Foo;
import org.osgi.framework.BundleContext;

```

```

import org.osgi.util.tracker.ServiceTracker;

/**
 * Simplest use of a ServiceTracker to get a service
 */
public class TrackingFooUser
{
 private ServiceTracker<Foo, Foo> serviceTracker;

 public TrackingFooUser(BundleContext bundleContext)
 {
 serviceTracker = new ServiceTracker<Foo, Foo>(bundleContext, Foo.class, null);
 serviceTracker.open();
 }

 public void doFoo() {
 Foo foo = serviceTracker.getService();
 //use foo
 //no need to return it... just don't use it for long.
 }

 public void shutdown() {
 serviceTracker.close();
 }
}

```

## Composing advanced features by using OSGi Declarative Services

Simple features can be controlled by using bundle activator classes and direct implementation of interfaces such as `ManagedService` and `ServiceTracker`. As relationships between bundles become more complex, it can be better to use facilities such as OSGi Declarative Services (DS) to decompose a feature into individual services. DS (sometimes known as the Service Component Runtime, or SCR) provides lifecycle and injection management of OSGi services.

### About this task

Organizing your feature logic as a set of declarative services has a number of advantages:

- Activation of the service (which includes loading the Java classes that provide the service) can be deferred until the service is used; allowing the server to start quickly and to keep resource use to a minimum.
- A reference to the service is placed into the service registry, even when the service has not been activated, so that dependencies on the service can be resolved.
- Dependencies on other services can be injected at runtime, and activation of the various services will be ordered based on such dependencies.
- A service can be deactivated and reactivated when its service properties change, if required.

Detailed information about use of OSGi Declarative Services is available from a number of online resources, including the OSGi Community Wiki.

This task provides simple descriptions of how to declare your services to DS, how to obtain references to other services, and how to manage configuration properties for each service.

### Declaring your services to OSGi Declarative Services

You can use a separate XML file to declare each service within a bundle.

## About this task

The Declarative Services (DS) support operates on declared components, each of which is defined by an XML file in the bundle. When a bundle containing component declarations is added to the framework, DS reads each component declaration and registers provided services in the service registry. DS then manages the lifecycle of the component: controlling its lifecycle based on a combination of declared attributes and satisfied dependencies.

The XML description of components allows DS to resolve service dependencies without requiring the component to be instantiated, or its implementation classes to be loaded. This facilitates late and lazy resource loading, which helps improve server startup and reduce runtime memory footprint.

The XML files that describe the components are listed in the MANIFEST.MF file of the bundle using the Service-Component header, and by convention are located in the /OSGI-INF directory of the bundle.

There are a number of tools that can be used to generate the required XML; the following examples show the XML itself.

This topic describes a simple OSGi bundle using XML to declare its components to DS.

## Procedure

1. Identify your component through its implementation class name.

```
<component>
 <implementation class="com.example.bundle.HelloComponent"/>
</component>
```

2. Declare the service by referencing the name of the interface that it provides. This is the name that will be published to the service registry by DS when the bundle is started.

```
<component>
 <implementation class="com.example.bundle.HelloComponent"/>
 <service>
 <provide interface="com.example.HelloService"/>
 </service>
</component>
```

3. Name the component. The component name also acts as the service “persisted identity”, or PID, which is used to associate configuration properties with the service. Configuration properties with a matching PID will be injected into the component on activation, and whenever the properties are updated.

```
<component name="HelloService">
 <implementation class="com.example.bundle.HelloComponent"/>
 <service>
 <provide interface="com.example.HelloService"/>
 </service>
</component>
```

**Note:** In Liberty, a user can add the following element to the `server.xml` configuration file, and the properties will be injected into the `HelloComponent` class.

```
<HelloService firstKey="firstValue" secondKey="secondValue" />
```

4. Package the XML file into the bundle.

For example, the XML file is at the location `OSGI-INF/HelloService.xml`, and you add a header to the bundle manifest `MANIFEST.MF` file so that DS can locate the file:

```
Service-Component: OSGI-INF/HelloService.xml
```

If multiple components are packaged in the same bundle, the corresponding XML files must be entered as a comma-separated list. For example:

```
Service-Component: OSGI-INF/HelloService.xml, OSGI-INF/GoodbyeService
```

5. The Java implementation of the `HelloService` component is as follows:

```

package com.example.bundle;

import com.example;
import org.osgi.service.component.ComponentContext;

/*
 * This class must be public and have a public default constructor for it to be
 * usable by DS. This class is not required to be exported from the bundle.
 */
public class HelloComponent implements HelloService {
 /**
 * Optional: DS method to activate this component. If this method exists, it
 * will be invoked when the component is activated. Best practice: this
 * should be a protected method, not public or private
 *
 * @param properties
 * : Map containing service & config properties
 * populated/provided by config admin
 */
 protected void activate(ComponentContext cContext,
 Map<String, Object> properties) {
 modified(properties);
 }

 /**
 * Optional: DS method to deactivate this component. If this method exists,
 * it will be invoked when the component is deactivated. Best practice: this
 * should be a protected method, not public or private
 *
 * @param reason
 * int representation of reason the component is stopping
 */
 protected void deactivate(ComponentContext cContext, int reason) {
 }

 /**
 * Optional: DS method to modify the configuration properties. This may be
 * called by multiple threads: configuration admin updates may be processed
 * asynchronously. This is called by the activate method, and otherwise when
 * the configuration properties are modified while the component is
 * activated.
 *
 * @param properties
 */
 public synchronized void modified(Map<String, Object> properties) {
 // process configuration properties here
 }

 /**
 * Service method defined by com.example>HelloService interface
 */
 public void sayHello() {
 System.out.println("Hello");
 }
}

```

## Enabling a service to receive configuration data

To enable a service to receive configuration data, you associate the service with a persisted identity, and code the service to receive the data. You can also provide descriptions and default values for this data, and make the labels and descriptions available in several languages.

## About this task

To enable a service to receive configuration data, there are a number of steps involved. Only associating the service with a configuration Admin persisted identity and coding the service to receive configuration properties are mandatory, and might be considered sufficient for embedded scenarios. The remaining steps improve the configuration experience for users.

The steps involved in enabling a service to receive configuration data are described in the following subtopics:

### Procedure

1. Associate the service with a Configuration Admin PID (persisted identity).
2. Code the service to receive the configuration properties during activation and when the configuration is modified.
3. Provide descriptions and default values for configuration metadata.
4. Provide translated strings for configuration property labels and descriptions.

### Associating a service with a persisted identity:

You associate a set of configuration properties with its consuming component as described in the OSGi Configuration Admin specification by using the the persisted identity (PID).

## About this task

The OSGi Configuration Admin specification provides a number of association mechanisms, of which the following are most commonly used in Liberty:

### Register an implementation of `org.osgi.service.cm.ManagedService` or `org.osgi.service.cm.ManagedServiceFactory` directly with the OSGi Configuration Admin service (CA)

This is most commonly used in low-level kernel bundles, where service management through OSGi Declarative Services (DS) or Blueprint is not available at bundle start time. The registration specifies the PID that identifies the configuration set to be received.

### Define a service to DS

This is the most common way for services in feature bundles to receive their configuration. The service name is used as the PID to associate configuration data. DS receives the configuration set from CA and passes it on to the defined service.

## Example

A service might be declared by using the following entry in the project \*.bnd file:

```
Service-Component: com.ibm.ws.transaction; \
 provide:='com.ibm.tx.config.ConfigurationProvider'; \
 immediate:='true'; \
 modified:='modified'; \
 implementation:=com.ibm.ws.transaction.services.JTMConfigurationProvider
```

This generates the following XML code, which can also be coded by the developer instead of using the **bnd** Service-Component entry:

```
<component name="com.ibm.ws.transaction" xmlns="http://www.osgi.org/xmlns/scr/v1.1.0"
 immediate="true" modified="modified">
 <implementation class="com.ibm.ws.transaction.services.JTMConfigurationProvider" />
</service>
 <provide interface="com.ibm.tx.config.ConfigurationProvider" />
</service>
 <property name="service.vendor" value="IBM" />
</component>
```

The component name, `com.ibm.ws.transaction` in this example, is used as the PID for the association of configuration data. If this component does not provide any metadata to describe its configuration, you can specify configuration properties for the component by using that PID in the `server.xml` file, or an included file, by defining an entry of the following form:

```
<com.ibm.ws.transaction made.up.property.key="47">
```

### What to do next

Code the service to receive the configuration properties during activation and when the configuration is modified.

### Coding the service to receive configuration properties:

Configuration properties are available through the `org.osgi.service.component.ComponentContext` object that is provided on the activation method.

### Before you begin

You must complete the task described in “Associating a service with a persisted identity” on page 1119.

### About this task

If properties are updated after activation has occurred, the method used for injection depends on the context that the service provides in its OSGi Declarative Services (DS) declaration.

Generally, it is best to declare a method that is to be used specifically for injection of updated properties by using the **modified** attribute on the service declaration. If a modified method is not available, DS deactivates and then reactivates the service with the new properties.

Deactivating and then activating a service can also cause dependent services to be recycled, and should be avoided unless specifically required. Using the **modified** attribute is the preferred way to receive configuration updates.

### Example

In the previous task, “Associating a service with a persisted identity” on page 1119, you defined a service to DS. The following are examples of activate and modified methods from the DS declaration described in that task.

```
private static Dictionary<String, Object> _props = null;

protected void activate(ComponentContext cc) {
 _props = cc.getProperties();
}

protected void modified(Map<?, ?> newProperties) {
 if (newProperties instanceof Dictionary) {
 _props = (Dictionary<String, Object>) newProperties;
 } else {
 _props = new Hashtable(newProperties);
 }
}
```

When you get values from the configuration properties, use the following mechanisms to allow some flexibility:

- Code the methods to expect at least the default properties that are included in the same bundle, but to make allowances for user overrides, so that migration of user configuration is not necessary.



- Ignore redundant or unrecognized properties.

The service must be able to operate on the default configuration alone. To provide a reasonable level of function, user overrides must not be mandatory.

### What to do next

Provide descriptions and default values for configuration metadata

## Advanced Configuration

Advanced configuration includes information about providing descriptions and the default values for configuration and OSGi Metatype Service Extensions.

The configuration properties for each service can be described in metadata that complies with the OSGi Metatype Service specification. The resulting XML file is packaged into the bundle in the OSGI-INF/metatype directory, in accordance with the specification. For more information, see Providing Descriptions and default values for configuration.

The Liberty runtime and developer tools recognize some extensions to the OSGi Metatype specification for more complex configurations and a better presentation in a user interface. For more information, see OSGi Metatype Service Extensions.

### Describing configuration by using the OSGi Metatype service

The configuration properties for each service can be described in metadata that complies with the OSGi Metatype Service specification. The metadata can include default values, translatable names and descriptions, and information to allow validation of input values. The resulting XML file is packaged into the bundle that contains your service, in the OSGI-INF/metatype directory, in accordance with the specification.

### About this task

Providing metadata to describe your configuration is optional, but it does provide the following benefits:

- default values can be separated from the implementation code into the metatype XML file where they are easy to locate;
- appropriate data types and other validation data can be specified for each attribute, allowing validation by the configuration parser and developer tools, and simplifying the code you write to process the attributes;
- your configuration will be included in the XML schema that describes the available configuration to the developer tools and other utilities;
- translatable names and descriptions can be provided for each attribute, and will be displayed in the developer tools.

### Procedure

1. Create an xml file in the OSGI-INF/metatype directory of your bundle and add a namespace declaration for the OSGi Metatype namespaces:

```
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0">
</metatype:MetaData>
```

2. Add an object class definition (OCD) element to contain the set of attributes, with an identifier and, optionally, a name and description. Also provide a Designate element to map the OCD to the PID used in your code and the server.xml file.

```
<OCD name="b2c" description="bundle two config" id="b2c-id">
</OCD>
```

```
<Designate pid="testBundleTwo">
<Object ocdref="b2c-id" />
</Designate>
```

3. Add attribute definition (AD) elements for each configuration property, within the OCD. Each attribute needs an identifier which is also used in the `server.xml` file and in the code that receives the injected configuration. It can optionally have a name and description which can be used by the developer tools and other graphical tools. Specifying a data type allows the runtime environment to validate the input for that type and simplifies your processing code. Specifying a useful default value allows the user-supplied configuration to be minimal, and contains all your configuration defaults in a known location:

```
<AD name="boolProperty" description="a boolean property" id="boolProp"
type="Boolean" default="false" />
```

4. Then you have the following `metatype.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0" >

<OCD name="b2c" description="bundle two config" id="testBundleTwo-2-id">
 <AD name="textProperty" description="a text property"
 id="textProp" type="String" default="default string" />
 <AD name="boolProperty" description="a boolean property"
 id="boolProp" type="Boolean" default="false" />
 <AD name="intProperty" description="an integer property"
 id="intProp" type="Integer" default="14" />
</OCD>

<Designate pid="testBundleTwo-2">
 <Object ocdref="testBundleTwo-2-id" />
</Designate>

</metatype:MetaData>
```

5. Code your service to receive the configuration properties. Without the metatype description, all your properties will be provided as `String` values and will be processed as follows:

```
String textProp = (String) properties.get("textProp");
Boolean boolProp = Boolean.parseBoolean((String) properties.get("boolProp"));
int intProp = Integer.parseInt((String) properties.get("intProp"));
```

```
String textProp = (String) properties.get("textProp");
Boolean boolProp = (Boolean) properties.get("boolProp");
int intProp = (Integer) properties.get("intProp");
```

And the runtime environment will already have validated that the input values are of the correct types.

## What to do next

Provide translated strings for configuration property labels and descriptions.

## Single versus multiple configuration instances

You can also configure multiple configuration instances by using the OSGi metatype services.

As described in the “Describing configuration by using the OSGi Metatype service” on page 1121, you can use OSGi metatype service to support a single set of configuration properties for a given service (as identified by a configuration PID). It is also common to support multiple instances of the same configuration type, for example in the way that the Liberty profile supports multiple entries for applications and data sources. This can be done by providing a metatype definition that tells the Liberty configuration parser, and the Configuration Admin service, that it is dealing with a factory configuration.

Also, the class that receives the configuration needs to implement the `org.osgi.service.cm.ManagedServiceFactory` interface.

To support multiple instances of top-level configuration elements in the `server.xml` file as follows:

```
<server>
 <teenager name="joy" age="15" />
 <teenager name="angela" age="18" />
</server>
```

You must define the configuration in metadata by adding a **factoryPid** attribute to the `Designate` element.

**Note:** A **pid** attribute is still needed if you use a `ManagedServiceFactory` interface to receive the configuration; if you use a declarative service (DS) component, this is not required.

```
<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
 xmlns:ibm="http://www.ibm.com/xmlns/appservers/osgi/metatype/v1.0.0">

 <OCD id="teenager-ocd" name="teenager" >
 <AD id="name" name="name" type="String" />
 <AD id="age" name="age" type="Integer" />
 </OCD>

 <Designate factoryPid="teenager" pid="teenager">
 <Object ocdref="teenager-ocd" />
 </Designate>

</metatype:MetaData>
```

The `ManagedServiceFactory` implementation is registered as a `ManagedServiceFactory` service type, with the factory pid as follows:

```
bundleContext.registerService(ManagedServiceFactory.class, new MgdSvcFactoryImpl(), new Hashtable());
defaults.put(org.osgi.framework.Constants.SERVICE_PID, "teenager");
```

The `ManagedServiceFactory` implementation receives a set of properties for each instance of the *teenager* configuration, each one uniquely identified by its own (internally generated) PID which is provided to the `updated()` method as follows:

```
public void updated(String pid, Dictionary<String, ?> properties)
 throws ConfigurationException {
 String name = (String) properties.get("name");
 Integer age = (Integer) properties.get("age");
}
```

If a particular configuration instance is deleted, for example because one of the *teenager* elements is deleted from the `server.xml` file, the `ManagedServiceFactory` implementation is notified through the `deleted()` method, and the pid of the deleted instance is provided. This allows the `ManagedServiceFactory` implementation to keep track of the instances that are valid at any given time.

### Providing default instances of factory configurations:

You can create default instances of your factory configuration when using the OSGi metatype services. One of the design principles of Liberty is to keep user configuration as small and simple as possible. By providing default instances of your factory configurations, you don't have to add these configurations into the `server.xml` file.

## Example

To provide a default configuration instance, you need to include it in an XML file within your OSGi bundle, and reference the file by using the `IBM-Default-Config` header in the bundle manifest file as follows:

```
IBM-Default-Config: OSGI-INF/wlp/defaultInstances.xml
```

The format of the XML file is the same as that of the `server.xml` file, but you must specify a unique identifier for each instance. For example, to provide a default instance of the *teenager* configuration that is used in the example on the “Single versus multiple configuration instances” on page 1122 topic, the `defaultInstances.xml` file must have the following settings:

```
<server>
 <teenager id="predefined-teen1" name="Susie" age="19" />
</server>
```

The default instance is not exposed to users through the configuration schema, and therefore it is not visible in the development tools; however, you can document the instance so that your users can override the individual attributes in their `server.xml` files as follows:

```
<teenager id="predefined-teen1" age="13" />
```

This line of code will override the **age** attribute of the default instance, but the **name** attribute remains valid.

## Extensions to the OSGi metatype service

The Liberty runtime environment and developer tools recognize some extensions to the OSGi metatype specification for more complex configurations and a better presentation in a user interface.

### Runtime metatype extensions

Add this namespace to your `metatype.xml` file to use the following extensions:

```
xmlns:ibm="http://www.ibm.com/xmlns/appservers/osgi/metatype/v1.0.0"
```

#### **ibm:alias**

The alias extension is used to define a user-friendly name for the configuration while reducing the risk of clashes in the names of configuration elements in the `server.xml` file.

The following example shows the `ibm:alias` extension:

```
<OCD id="com.ibm.ws.jdbc.dataSource.properties"
 name="%properties"
 description="%properties.desc"
 ibm:alias="properties">
 <AD id="username".../>
</OCD>
```

In this example `properties` is the user-friendly name for the configuration. The alias must be different from the `id`.

When the `ibm:alias` entry is used in the `server.xml` file, it must be prefixed with the product extension name. The product extension name for extensions that are installed in the default user location is `usr`. For product extensions defined to the Liberty installation by using an `extension-name.properties` file in the `wlp/etc/extension` directory, the product extension name is the name that is chosen for `extension-name`.

For the metatype shown in the previous example, if the feature is installed in the default `usr` location then the following are examples of valid `server.xml` entries:

```
<usr_properties username="JANE"/>
<com.ibm.ws.jdbc.dataSource.properties username="JANE"/>
```

## ibm:type

Standard attribute types are defined in the metatype specification. Several IBM extended types are available. For more information, see “Extended types.”

## ibm:reference

The reference attribute specifies the OCD type that a PID references. It is used only with the `ibm:pid` type and supports nesting of elements in the `server.xml` file; see “Nesting configuration elements” on page 1128.

The following example shows the `ibm:reference` extension: `<AD id="fooRef" type="String" ibm:type="pid" ibm:reference="com.ibm.ws.foo".../>`

## ibm:final

The final attribute indicates that the value cannot be specified in the config. Instead, the default value from the metatype is always used. Use `name="internal"` to indicate that tools not display this property.

The following example shows the `ibm:final` extension: `<AD id="foo" name="internal" ibm:final="true" type="String" default=${someVariable}"/>`

## ibm:variable

The variable attribute is used to specify a variable to be used for the default value if one is not specified. The behavior is to choose, in order:

- The value that is specified in `server.xml`
- The value that is specified as a system property, for example in `bootstrap.properties`
- The default value from the metatype

The following example shows `ibm:variable`: `<AD id="traceString" ibm:variable="trace.string" default=*.all=enabled".../>`

## ibm:unique

The unique attribute indicates that a configuration value must be unique across all attribute definitions that use the same unique attribute group. The following unique attribute groups are supported:

- `jndiName`: use this group on attributes that register a service by using the `osgi.jndi.service.name` property with the JNDI name. For more information, see “Developing with the JNDI default namespace in a Liberty feature” on page 1139

## default value syntax

You can use `${prop-name}` syntax in default expressions to construct strings out of other configuration properties.

The following example shows a default value syntax:

```
<AD id="httpEndpoint.target"
 name="internal" description="internal use only"
 ibm:final="true" required="false" type="String"
 default="(&#x28;virtualHost=${id}) (enabled=true)"/>
```

## Extended types

### Duration

The duration type is used to express a time. It is described in multiple units of time. For example, "1h30m" would be an hour and a half. "1d5h10s" would be 1 day, 5 hours, and 10 seconds. The units are globalized, so users enter the values by using abbreviations from their local language.

For English, the following list shows the available units:

- d - Days
- h - Hours
- m - Minutes
- s - Seconds
- ms - Milliseconds

By default, when using the type duration, the value that is specified by the user is evaluated in milliseconds. For example, "10s" would be a long value of 10000 in the dictionary. Furthermore, if a user specifies a value without any unit, this value will be evaluated in milliseconds. For example, a value of "10" would be evaluated as 10 milliseconds. However, you can also specify the duration type such that it evaluates into a different unit. For example, specifying a value of "10" with `ibm:type="duration(s)"` will be evaluated as 10 seconds, and stored as 10 in the dictionary.

The following list shows the possible types:

- duration(h)
- duration(m)
- duration(s)
- duration(ms)
- duration

There is no difference between specifying duration and duration(ms).

**Note:**

Best practice: Always include a unit in the value, and express the value with the unit that is easiest to read. For example, instead of specifying a value of "7200" with `ibm:type="duration(s)"`, specify the value as "2h".

The following examples show the duration type:

- `<AD id="timeout" type="String" ibm:type="duration(s)".../>`
- `<AD id="timeout" type="String" ibm:type="duration".../>`

**Location**

The location type allows UI tools to provide a more helpful presentation of attributes that represent various file and directory locations. It does not affect processing by the runtime environment. The dictionary object is always a String.

The following examples show the possible types:

**location**

References a file. The reference can be an absolute, relative file, or it can be a url to a file.

**location(file)**

References a file by using an absolute or relative file path.

**location(dir)**

References a dir by using an absolute or relative file path.

**location(url)**

References a file at the end of a url.

The following example shows the location type:  
`<AD id="location" name="%appmgr.location.name" description="%appmgr.location.desc" type="String" required="true" ibm:type="location"/>`

**Password**

The password type is used for password fields. When used, the dictionary object is an instance of `com.ibm.wsspi.kernel.service.utils.SerializableProtectedString`. The value of the password field is not logged in the trace file. Developer tools displays the encoding options that can be used for a password field. Valid encoding options are xor and aes.

The following example shows the password type:`<AD id="password" type="String" ibm:type="password".../>`

### Hashed password

The passwordHash type is similar to the password type and is used for hashed password fields. When used, the dictionary object is an instance of `com.ibm.wsspi.kernel.service.utils.SerializableProtectedString`. The value of the hashed password field is not logged in the trace file. Developer tools displays the encoding options that can be used for a hashed password field. Valid encoding options are xor, aes, and hash.

Validate a new password against a hashed password by using the `PasswordUtil.encode(String, String, Map)` method, with the following parameters:

1. New password.
2. Hash algorithm, which is obtained by calling the `PasswordUtil.getCryptoAlgorithm` method. The hash algorithm must match the algorithm of the hashed password.
3. Properties object, where one of the properties uses `PasswordUtil.PROPERTY_HASH_ENCODED` for the key and the hashed password for the value.

If the return value of `PasswordUtil.encode` is the same as the hashed password, then the passwords match.

The following example shows the passwordHash type:`<AD id="hashedPassword" type="String" ibm:type="passwordHash".../>`

### Pid

The pid type is used to reference another object in the config. It is used with the `ibm:reference` attribute and supports nesting of elements in the `server.xml`; see “Nesting configuration elements” on page 1128.

The following example shows the pid type:`<AD id="fooRef" type="String" ibm:type="pid" ibm:reference="com.ibm.ws.foo".../>`

### OnError

The onError type results in an instance of the onError enumeration in the dictionary. The possible values are WARN, FAIL, and IGNORE.

The following example shows the onError type:`<AD id="errorBehavior" type="String" ibm:type="onError".../>`

## User interface metatype extensions

Add this namespace to your `metatype.xml` file to use the following extensions:

```
xmlns:ibmui="http://www.ibm.com/xmlns/appservers/osgi/metatype/ui/v1.0.0"
```

### ibmui:localization

The localization extension is used to specify the metatype localization file. The metatype localization file is used to look up the translations for labels and descriptions of other UI extensions. In most cases, the value of the `ibmui:localization` extension matches the localization attribute on the `<Metadata>` element.

The following example shows the `ibmui:localization` extension:

```

<OCD id="com.ibm.ws.jdbc.dataSource.properties"
 name="%properties"
 description="%properties.desc"
 ibmui:localization="OSGI-INF/110n/metatype">
 <AD id="username".../>
</OCD>

```

### ibmui:extraProperties

The extraProperties extension is used to indicate that an arbitrary set of configuration attributes can be set on this configuration.

The following example shows the ibmui:extraproperties extension:

```

<OCD id="com.ibm.ws.jdbc.dataSource.properties"
 name="%properties"
 description="%properties.desc"
 ibmui:extraProperties="true">
 <AD id="username".../>
</OCD>

```

The label and description that is associated with extension is looked up in the metatype localization file (if one is specified by using the ibmui:localization extension). For the extension label, first extraProperties.<ocd id>.name and then extraProperties.name keys are checked. For the extension description, first extraProperties.<ocd id>.description and then extraProperties.description keys are checked.

### ibmui:group

The group extension is used to specify that the attribute belongs to a group. In the user interface, the attributes that are annotated with the same group are grouped.

The following examples show the ibmui:group extension:

- <AD id="username" ibmui:group="userInfo".../>
- <AD id="password" ibmui:group="userInfo".../>
- <AD id="port" ibmui:group="hostInfo".../>

The group label and description information are looked up in the metatype localization file (if one is specified by using the ibmui:localization extension). For the group label, first <group>.<ocd id>.name and then <group>.name keys are checked. For the group description, first <group>.<ocd id>.description and then <group>.description keys are checked.

### Nesting configuration elements:

You can use metatype extensions to define configuration that can be expressed as nested XML elements in the server.xml file.

### Examples

The following example shows how to support this user configuration in the server.xml file:

```

<family mother="jane" father="john">
 <child name="susie" age="8" />
 <child name="danny" age="5" />
</family>

```

The metatype XML uses ibm:type="pid" and ibm:reference as shown in following example:

```

<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData
 xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
 xmlns:ibm="http://www.ibm.com/xmlns/appservers/osgi/metatype/v1.0.0">

 <OCD id="family" name="family">

```



```

 <AD id="mother" name="mother" type="String" default="Ma" />
 <AD id="father" name="father" type="String" default="Pa" />
 <AD id="child" name="child" ibm:type="pid" ibm:reference="child-pid"
 required="false" type="String" cardinality="6" />
</OCD>

<Designate pid="family">
 <Object ocdref="family" />
</Designate>

<OCD id="child" name="child" >
 <AD id="name" name="name" type="String" />
 <AD id="age" name="age" type="Integer" />
</OCD>

<Designate factoryPid="child-pid">
 <Object ocdref="child" />
</Designate>

</metatype:MetaData>

```

The following example show how the code that receives the family properties uses the ConfigurationAdmin service to obtain the child property sets:

```

public void updated(Dictionary<String, ?> properties)
 throws ConfigurationException {

 Set<String> pids = new HashSet<String>();
 String mother = "null";
 String father = "null";

 try {
 if (properties != null) {
 mother = (String) properties.get("mother");
 father = (String) properties.get("father");
 String[] children = (String[]) properties.get("child");
 if (children == null || children.length == 0) {
 return;
 }

 // Get the configuration admin service
 ConfigurationAdmin configAdmin = null;
 ServiceReference configurationAdminReference =
 bundleContext.getServiceReference(ConfigurationAdmin.class.getName());

 if (configurationAdminReference != null) {
 configAdmin = (ConfigurationAdmin)
 bundleContext.getService(configurationAdminReference);
 }

 for (String childPid : children) {
 pids.add(childPid);
 Configuration config = configAdmin.getConfiguration(childPid);
 String name = (String) config.getProperties().get("name");
 Integer age = (Integer) config.getProperties().get("age");
 }
 }
 }

 catch (Exception e) {
 e.printStackTrace();
 }
}

```

## Localizing the configuration metadata

The name and description attributes of each metadata entry can be localized, and the translated strings packaged into language-specific properties files.

### Example

The following example shows how the location of the localized files is specified in the header of the metatype file:

```
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
 localization="OSGI-INF/I10n/metatype">
```

where OSGI-INF/I10n is the location of the translated properties files in the bundle, and metatype is the prefix of the default language properties file. For example, if the default values, usually in English, are in a file called metatype.properties, then each locale is added with its own suffix: metatype\_fr.properties, metatype\_es.properties and so on.

Unlike the metatype XML file, which must always be in the OSGI-INF/metatype directory, the translated files can be in any location that is within the bundle and specified by the **localization** attribute. It is better to not put the properties files in the OSGI-INF/metatype directory alongside the metatype XML file; the Metatype service attempts to parse anything in that location as an XML file, and although that does not cause a failure, it generates unwanted exceptions in the console. The Liberty convention is to put them in the OSGI-INF/I10n directory, but that is not mandatory.

In the metatype XML file, to show that a value is a localized string you use a percent sign at the start of the value. For example, you might use the following definition in the metatype XML file:

```
<AD name="%client.inactivity.timeout" description="%client.inactivity.timeout.desc"
 id="clientInactivityTimeout" required="false" type="Integer" default="60" />
```

and you might use the following definition in the properties file:

```
client.inactivity.timeout=Client inactivity timeout
client.inactivity.timeout.desc=The maximum duration, in seconds, between transactional requests
from a remote client. Any period of client inactivity that exceeds this timeout results in the
transaction being rolled back in this application server.
```

## Providing an application endpoint

You can make a Liberty feature available as web applications by including one or more web application bundles (WABs) in the feature. A WAB is an OSGi bundle with a Web-ContextPath manifest header.

### Procedure

To enable WABs in your feature's bundles, add the com.ibm.wsspi.appserver.webBundle-1.0 feature to the Subsystem-Content: header in the .mf file of your feature:

```
Subsystem-Content:
my.user.feature.bundle; version="[1,1.0.100)",
com.ibm.wsspi.appserver.webBundle-1.0; type="osgi.subsystem.feature"
```

## Securing an application endpoint

You can secure your feature's application endpoint by performing the following steps:

### Procedure

1. In the .mf file of your feature, add the com.ibm.wsspi.appserver.webBundleSecurity-1.0 feature to the Subsystem-Content: header. This addition causes any protected servlets (as specified in your feature bundle's WEB-INF/web.xml file) to be authenticated, and enables role base authorization. You can also assign users, groups, and special subjects to any roles that are defined in the WEB-INF/web.xml file.

```
Subsystem-Content:
my.user.feature.bundle; version="[1,1.0.100)",
com.ibm.wsspi.appserver.webBundleSecurity-1.0; type="osgi.subsystem.feature"
```

2. To map roles to users, groups, and special subjects, do the following steps:
  - a. Add the IBM-Authorization-Roles header to your OSGi bundle's MANIFEST.MF file. The header must specify a name which is the id of a role mapping you specify in the server.xml file.  
IBM-Authorization-Roles: my.feature.role.map
  - b. In the server.xml file, add an authorization-roles element to map the role names to users and groups. The id attribute of the authorization-roles element must have the same value as the IBM-Authorization-Roles header in the MANIFEST.MF file. Add a <security-role> subelement for each role that you want to assign user and groups to.

```
<authorization-roles id="my.feature.role.map">
 <security-role name="employee">
 <special-subject type="ALL_AUTHENTICATED_USERS"/>
 </security-role>
 <security-role name="manager">
 <user name="bob"/>
 <user name="mary"/>
 <group name="managers"/>
 </security-role>
</authorization-roles>
```

## Liberty SPI utilities

Liberty provides service programming interfaces (SPI) to complete various tasks.

### Resource location symbols

Liberty user configuration is made more portable through the use of variables that represent symbolic locations. Use of these variables helps to prevent the coding of absolute paths that would make the user configuration brittle and less portable. Feature code that receives configuration properties might have to deal with values that contain such variables.

The location service of Liberty can be used to resolve symbolic locations to physical resources. For example, the symbolic location `${wlp.install.dir}/myFile` can be mapped to the local file `myFile` in the installation directory of Liberty. Most methods return a `WsResource` object that wraps the physical resource, but you can also use the `resolveString` method to transform the symbolic location into a `String` that can be used to obtain a `File` object.

The name of the location service is `com.ibm.wsspi.kernel.service.location.WsLocationAdmin` and it is provided by the Liberty kernel, so you do not have to specify a feature in your `server.xml` file to make it available. The Java API documentation for each Liberty SPI is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate `.zip` file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

### Symbols

The `com.ibm.wsspi.kernel.service.location.WsLocationConstants` class defines symbols that refer to directory locations:

- /
- server.config.dir
- server.output.dir
- server.workarea.dir
- shared.app.dir
- shared.config.dir
- shared.resource.dir

- wlp.install.dir
- wlp.server.name
- wlp.user.dir
- usr.extension.dir

For the meaning of each symbol, see “Directory locations and properties” on page 894.

## Monitoring local files for changes

Liberty has highly dynamic behavior, responding to changes in configuration, applications and other resources. Much of this dynamic behavior is based on monitoring of the local file system for changes. The service that performs this monitoring is available to all Liberty features through the FileMonitor SPI. The file monitor service is provided by the Liberty kernel, so you do not have to specify a feature in your server.xml file to make it available.

### About this task

The FileMonitor SPI provides different properties to specify what resources are monitored and with what frequency. You have to implement the FileMonitor interface and register the implementation class into the service registry.

The Java API documentation for each Liberty SPI is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `{wlp.install.dir}/dev` directory.

### Example

```
...
import com.ibm.wsspi.kernel.filemonitor.FileMonitor;
...

public class MyFileMonitor implements FileMonitor {
 ...

 private final BundleContext bundleContext;
 ...

 public MyFileMonitor(BundleContext bundleContext) {
 this.bundleContext = BundleContext;
 ...
 }

 public ServiceRegistration<FileMonitor> monitorFiles(Collection<String> paths, long monitorInterval) {
 ...
 final Hashtable<String, Object> fileMonitorProps = new Hashtable<String, Object>();
 fileMonitorProps.put(FileMonitor.MONITOR_FILES, paths);
 fileMonitorProps.put(FileMonitor.MONITOR_INTERVAL, monitorInterval);
 ...
 return bundleContext.registerService(FileMonitor.class, this, fileMonitorProps);
 }
 ...
}
```

## Configuring tracing and logging for features in the Liberty profile

You can use the tracing and logging mechanism of the Liberty profile for Liberty features. The logging service is part of the Liberty kernel so you do not have to specify a feature in your server.xml file to use it.

### About this task

Liberty provides the following SPIs for integrating tracing and logging in your customized feature code:

#### **com.ibm.websphere.ras**

The `com.ibm.websphere.ras` package provides classes to log messages and trace records, as well as some extension points. In general, feature code can use the `java.util.logging` package to log

trace and messages, and to control the output through Liberty logging configuration, but the extended capability of the WebSphere package is sometimes useful and the trace guards are slightly more efficient when trace is disabled.

#### **com.ibm.websphere.ras.annotations**

The `com.ibm.websphere.ras.annotations` package provides annotations for use with classes in the other packages. For example, an `@Sensitive` annotation can be used to prevent the contents of the annotated variable from appearing in trace or message output.

#### **com.ibm.ws.ffdc**

The `com.ibm.ws.ffdc` package provides facilities to write first failure data capture (FFDC) records to assist in debugging unexpected exceptions.

#### **com.ibm.wsspi.logging**

The `com.ibm.wsspi.logging` package provides interception points of log and ffdc records.

The Java API documentation for each Liberty SPI is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

## **Procedure**

The following steps show you how to configure an example Liberty feature, called `myfeature`, to use the tracing and logging mechanism of Liberty:

1. Specify the location of the message file for the feature `myfeature`, and the name of the group that is required by the `com.ibm.websphere.ras.TraceComponent` class.

```
import java.util.ResourceBundle;

public class myFeatureConstants {

 public static final String TR_RESOURCE_BUNDLE =
 "com.mycompany.myFeature.internal.resources.FeatureMessages";

 public static final String TR_GROUP = "myFeature";

 public static final ResourceBundle messages = ResourceBundle.getBundle(TR_RESOURCE_BUNDLE);

}
```

2. In the implementation class of the feature service code, call the `register()` method of the `com.ibm.websphere.ras.TraceComponent` class to register the implementation class with the trace manager that is provided by Liberty. Then, you can configure the trace manger to track the DS methods of the feature.

```
...
import com.ibm.websphere.ras.Tr;
import com.ibm.websphere.ras.TraceComponent;

public class myFeatureServiceImpl {

 private static final TraceComponent tc = Tr.register(myFeatureServiceImpl.class);

 protected void activate(ComponentContext cc, Map<String, Object> newProps) {
 if (tc.isDebugEnabled()) {
 Tr.debug(tc, "myFeatureComponentImpl activated"); }
 }

 ...
}
```

3. Use the `TraceOptions` annotation to specify the trace group name and the message bundle name.

```
@TraceOptions(traceGroup = myFeatureConstants.TR_GROUP, messageBundle =
 myFeatureConstants.TR_RESOURCE_BUNDLE)
package com.mycompany.myFeature;
```

```
import com.ibm.websphere.ras.annotation.TraceOptions;
import com.mycompany.myfeature.internal.myFeatureConstants;
...
```

## Generating first failure data capture (FFDC) records:

FFDC records include the exception stack and optional additional data that is recorded when an unexpected exception is caught by your code. Methods on the `com.ibm.ws.ffdc.FFDCFilter` class are used to generate these records, and there are a number of methods that might cause a variety of data to be captured.

### Example

A typical use of the `FFDCFilter` class is as follows:

```
try{
 // ... do something
} catch (Exception e) {
 FFDCFilter.processException(e, getClass().getName(), unique-probe-id);
 if (TraceComponent.isAnyTracingEnabled() && tc.isDebugEnabled()) {
 Tr.debug(tc, "Exception when doing something; " + e);
 }
 return;
}
```

Where the source id (class name in this example) and the unique probe id (typically the source code line number) combine to provide the exact location in the source code that generates the resulting record. By default, the records are written to the `/${server.output.dir}/logs/ffdc` directory.

The file space used by the FFDC records in the case of a persistently occurring exception is limited by automatic filtering of duplicate records. For any matching source id, probe id, and exception name, at most 10 exceptions with unique messages are written per day.


Feature code can contribute data to FFDC records by registering a `com.ibm.ws.ffdc.DiagnosticModule` implementation with the FFDC class. Feature code can also intercept FFDC records by registering a `com.ibm.wsspi.logging.IncidentForwarder` implementation with the FFDC class.




## Adding web services global handlers


8.5.5.4

Components that need to register web services handlers to all the web services end points must implement the `Handler` interface and register that implementation in the service registry.

### Before you begin

The global handler service is provided either by **jaxws-2.2**, **jaxrs-1.1**,  **jaxrs-2.0**, or **jaxrs-2.0 client**, so you must specify the following feature or feature combinations in your `server.xml` file:

- **jaxws-2.2**
- **jaxrs-1.1**
-  **8.5.5.6** **jaxrs-2.0**
-  **8.5.5.6** **jaxrs-2.0 client**
- **jaxws-2.2** and **jaxrs-1.1**
-  **8.5.5.6** **jaxws-2.2** and **jaxrs-2.0**

-  **8.5.5.6** **jaxws-2.2** and **jaxrs-2.0 client**

## About this task

The Handler SPI provides different properties to specify the `ENGINE_TYPE`, the `FLOW_TYPE`, and the client side (`IS_CLIENT_SIDE`) or the server side (`IS_SERVER_SIDE`) where handlers take effect.

You must implement the Handler interface and register the implementation class into the service registry.

The Java API documentation for each Liberty SPI is available in a separate compressed file in one of the Javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

## Deploying the handler bundle:

You can deploy the handler bundle by using the WebSphere Application Server Developer Tools for Eclipse .

## Procedure

1. Click **File** > **New** > **Other** and then expand **OSGi**.
2. Click **OSGi Bundle Project** and click **Next**. The New OSGi Bundle Project window opens.
3. Enter MyHandler as the Project name. In the Target runtime list, select **WebSphere Application Server Liberty**. If no runtime exists, click **New Runtime** to create a WebSphere(r) Application Server Liberty runtime.
4. Clear the **Add bundle to application** ratio.
5. Click **Next** twice and go to the OSGi Bundle page.
6. On the OSGi Bundle page, check **Generate an activator, a Java class that controls the life cycle of the bundle**. Leave the **Activator name** as `myhandler.Activator` and click **Finish**.
7. Click **Window** > **Preferences** > **Plug-in Development** > **Target Platform** and select **WebSphere Application Server Liberty with SPI**.

**Note:** Ensure that you have added WebSphere Application Server Liberty runtime in 3.

8. Click **Apply** and click **OK**.
9. Expand **MyHandler** > **BundleContent** > **META-INF** and open the `MANIFEST.MF` file by using the **Plug-in Manifest Editor**.
10. Create the MyHander and MyActivitor classes:

```
...
import com.ibm.wsspi.webservices.handler.Handler;
...

public class MyHandler implements Handler {
 ...
 public void handleFault(GlobalHandlerMessageContext arg0) {
 ...
 }
 public void handleMessage(GlobalHandlerMessageContext msgctxt) throws Exception {
 if (msgctxt.getFlowType().equalsIgnoreCase(HandlerConstants.FLOW_TYPE_OUT)) {
 }
 ...
 }
 ...
}

public class MyActivator implements BundleActivator {
 ...
 public void start(BundleContext context) throws Exception {
```

```

 final Hashtable<String, Object> handlerProps = new Hashtable<String, Object>();
 handlerProps.put(HandlerConstants.ENGINE_TYPE, HandlerConstants.ENGINE_TYPE_JAXWS);
 handlerProps.put(HandlerConstants.FLOW_TYPE, HandlerConstants.FLOW_TYPE_IN);
 handlerProps.put(HandlerConstants.IS_CLIENT_SIDE, true);
 handlerProps.put(HandlerConstants.IS_SERVER_SIDE, true);
 handlerProps.put(org.osgi.framework.Constants.SERVICE_RANKING, 3);
 MyHandler myHandler = new MyHandler();
 context.registerService(Handler.class, myHandler, handlerProps);
 ...
 }
 ...
}

```

11. Click **File > New > Other** and then expand **OSGi**.
12. Click **Liberty Feature Project** and then click **Next**. The Liberty Feature Project window opens.
13. Specify **MyHandlerFeature** as the Project name.
14. In the Target runtime list, select **WebSphere Application Server Liberty** and click **Next**. The OSGi Bundles Selection Page opens.
15. On OSGi Bundles Selection Page, select **MyHandler 1.0.0** as the **Contained Bundles** and click **Finish**.
16. Modify **Manifest:MyHandler** in **MyHandler** project. Click the **MANIFEST.MF** tab and add **com.ibm.wsspi.webservices.handler** to the **Import-pacakge** element.
17. Right-click the **MyHandlerFeature** project, click **Install Feature** to install the feature to Liberty runtime.
18. Edit the **server.xml** file to enable the **MyHandlerFeature**:

```

<featureManager>
<feature>jsp-2.2</feature>
<feature>jaxws-2.2</feature> // you can also use one of the following feature or feature combinations: jaxrs-1.1, jaxrs-2.0,
jaxws-2.2 and jaxrs-1.1, jaxws-2.2 and jaxrs-2.0, jaxws-2.2 and jaxrsClient-2.0
<feature>usr:MyHandlerFeature</feature>
</featureManager>

```

## Exposing REST endpoints within Liberty

8.5.5.4

You can use the REST Handler framework in the Liberty SPI to expose new REST endpoints.

### About this task

The REST Handler framework is for Liberty extenders to use when exposing new REST endpoints. You can expose REST endpoints in an OSGi component, or a set of components.

### Procedure

1. Create an OSGi component that registers itself as listening to a sub-root that appends to **/ibm/api** and implements the **com.ibm.wsspi.rest.handler.RESTHandler** interface; for example:

```

@Component(service = { RESTHandler.class },
 configurationPolicy = ConfigurationPolicy.IGNORE,
 immediate = true,
 property = { "service.vendor=IBM",
 RESTHandler.PROPERTY_REST_HANDLER_ROOT + "/myTest/abc" })
public class RESTHANDLERTest1 implements RESTHandler {
 ...
}

```

2. Package the component into an OSGi bundle that is part of your extended user feature.
3. Ensure that your feature includes the OSGi subsystem content:

```
com.ibm.websphere.appserver.restHandler-1.0; type="osgi.subsystem.feature"
```
4. Configure SSL certificates in the **server.xml** file.
5. Configure a user or group to the administrator role in the **server.xml** file.



- Map to the administrator role for Liberty
6. Start your feature.
- Starting the feature starts the REST Handler framework and registers your OSGi component. After the feature starts, you can make calls to `https://<host>:<https_port>/ibm/api/myTest/abc`.

## Including protected features

Your feature can include one or more other features by listing them in the Subsystem-Content header of the feature manifest file. Any feature in the same product extension as your own feature can be included; if the included feature is in a different product extension, or in Liberty, it must have public or protected visibility.

The included feature must be specified by its Subsystem-SymbolicName, and have a type of "osgi.subsystem.feature"; for example:

```
Subsystem-Content:
 com.ibm.wsspi.appserver.webBundle-1.0; type="osgi.subsystem.feature",
 com.ibm.websphere.appserver.json-1.0; type="osgi.subsystem.feature"
```

For information on Liberty public features, see “Liberty features” on page 483. The following section describes Liberty protected features.

## Liberty protected features

### Application Manager

This feature provides advanced capability for implementing new application containers.

Subsystem-SymbolicName: com.ibm.websphere.appserver.appmanager-1.0.

Provided API and SPI:

- dev/api/ibm/com.ibm.websphere.appserver.api.basics\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.application\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar

### Classloader service

This feature provides advanced capability for implementing new application containers.

Subsystem-SymbolicName: com.ibm.websphere.appserver.classloading-1.0.

Provided API and SPI:

- dev/spi/ibm/com.ibm.websphere.appserver.spi.classloading\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar

### Container services

This feature provides advanced capability for implementing new application containers.

Subsystem-SymbolicName: com.ibm.websphere.appserver.containerServices-1.0,

Provided API and SPI:

- dev/spi/ibm/com.ibm.websphere.appserver.spi.containerServices\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.anno\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.javaeedd\_1.0.9.jar

### Transaction manager 1.1

This feature provides a JTA 1.1 compliant transaction manager.

Subsystem-SymbolicName: com.ibm.websphere.appserver.transaction-1.1.

Provided API and SPI:

- dev/api/spec/com.ibm.ws.javaee.transaction.1.1\_1.0.9.jar
- dev/api/ibm/com.ibm.websphere.appserver.api.transaction\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.containerServices\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.anno\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.javaeedd\_1.0.9.jar

### Transaction manager 1.2

This feature provides a JTA 1.2 compliant transaction manager.

Subsystem-SymbolicName: com.ibm.websphere.appserver.transaction-1.2.

Provided API and SPI:

- dev/api/spec/com.ibm.ws.javaee.transaction.1.2\_1.0.9.jar
- dev/api/ibm/com.ibm.websphere.appserver.api.transaction\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.containerServices\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.anno\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.javaeedd\_1.0.9.jar

### Web Bundle

This feature supports the use of web application bundles (WABs) in features. Include this feature if your feature provides an application endpoint, as described in “Providing an application endpoint” on page 1130.

Subsystem-SymbolicName: com.ibm.wsspi.appserver.webBundle-1.0.

Provided API and SPI:

- dev/api/spec/com.ibm.ws.javaee.servlet.3.0\_1.0.9.jar
- dev/api/ibm/com.ibm.websphere.appserver.api.servlet\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.servlet\_1.0.9.jar

### Web Bundle Security

This feature supports the application of security to web bundles; see “Securing an application endpoint” on page 1130.

Subsystem-SymbolicName: com.ibm.wsspi.appserver.webBundleSecurity-1.0.

Provided API and SPI:

- dev/api/spec/com.ibm.ws.javaee.servlet.3.0\_1.0.9.jar
- dev/api/ibm/com.ibm.websphere.appserver.api.servlet\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.servlet\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.containerServices\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.anno\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.artifact\_1.0.9.jar
- dev/spi/ibm/com.ibm.websphere.appserver.spi.javaeedd\_1.0.9.jar

## Locating OSGi applications

You can use classes in the `org.apache.aries.blueprint` package to extend the OSGi application programming model; this third-party SPI is provided through the `blueprint-1.0` server feature. You must access OSGi application bundles in to apply your extensions. In Liberty, OSGi applications run as Subsystems. To locate an OSGi application you can create a `ServiceTracker` in a user feature.

## About this task

This topic describes how the developer of a user feature can locate running OSGi applications. This task is often required for user features that provide programming model extensions to OSGi applications. For example, a new user feature might provide such extensions by implementing a new bundle extender, often referred to as a *container*, or, more simply, by tracking and invoking services published from within certain OSGi applications.

Such user features must use the `BundleContext` of a particular running OSGi application to create new `BundleTracker` and `ServiceTracker` instances. This `BundleContext` can be obtained from the `org.osgi.service.subsystem.Subsystem` that is associated with the OSGi application. The following procedure describes how to obtain that `SubSystem` service.

## Procedure

To locate an OSGi application by creating a `ServiceTracker` in a user feature, complete the following steps:

1. Construct an `org.osgi.framework.Filter` that is targeted to the `Subsystem` that you want to locate.
2. Create an `org.osgi.util.tracker.ServiceTracker` that uses the `Filter` from step 1 to obtain the `org.osgi.service.subsystem.Subsystem` service that is associated with the OSGi application that you want to locate. This `Subsystem` service instance provides everything you require to work with the OSGi application.

## Example

The following example shows how to locate an application with symbolic name `my.app` by using a `ServiceTracker` in a user feature:

```
import org.osgi.framework.BundleContext;
import org.osgi.service.subsystem.Subsystem;
import org.osgi.util.tracker.ServiceTracker;
import org.osgi.util.tracker.ServiceTrackerCustomizer;
```

In the following code extract, the variable `ctx` is the `BundleContext` of one of the bundles of the user feature:

```
String SERVICE_FILTER = "(objectClass=org.osgi.service.subsystem.Subsystem)
 (subsystem.type=osgi.subsystem.application)(subsystem.symbolicName=my.app))"

 org.osgi.framework.Filter filter = ctx.createFilter(SERVICE_FILTER);
```

The last 'null' parameter can be replaced with an instance of a class that implements `ServiceTrackerCustomizer<Subsystem, Subsystem>`:

```
org.osgi.util.tracker.ServiceTracker<Subsystem, Subsystem> str = new ServiceTracker<Subsystem, Subsystem>(ctx, filter, null);
```

The `SERVICE_FILTER` can be constructed to make use of such constants as:

```
org.osgi.framework.Constants.OBJECTCLASS;
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_SYMBOLICNAME_PROPERTY;
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_TYPE_APPLICATION;
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_TYPE_PROPERTY;
```

## Developing with the JNDI default namespace in a Liberty feature

You can make an object available in the default Java Naming and Directory Interface (JNDI) namespace. To do that, you must register it in the OSGi service registry with the `org.osgi.jndi.service.name` service property. The value of `org.osgi.jndi.service.name` is the required JNDI name. Similarly, to find an object in the default JNDI namespace, you can search the OSGi service registry with the `org.osgi.jndi.service.name` service property. The value of `org.osgi.jndi.service.name` is the JNDI name.

## About this task

Compared with explicitly calling `Context.bind` or `Context.lookup`, using the service registry has the following benefits:

- Your feature works properly when `jndi-1.0` is enabled, but your feature does not need an explicit dependency on JNDI.
- You do not need to explicitly unbind objects from JNDI when your feature is removed because the OSGi framework automatically unregisters services when bundles are stopped.
- You can easily implement lazy initialization using declarative services or `ServiceFactory` rather than using `Reference` and `ObjectFactory`.

For more information about JNDI, see [Naming](#).

## Procedure

1. Register a service using the `osgi.jndi.service.name` property with the JNDI name. For more information about registering services, see “[Registering OSGi services](#)” on page 1113.
2. Update your `metatype.xml` to allow a JNDI name to be specified in server configuration. To allow users to specify a JNDI name for your service, you should use the `jndiName` id for consistency with other features in the Liberty run time, for example:

```
<AD id="jndiName" name="JNDI name" description="JNDI name for a widget." type="String" ibm:unique="jndiName"/>
```

You can use an internal attribute to automatically set the `osgi.jndi.service.name` service property with the value of the `jndiName` attribute, for example:

```
<AD id="osgi.jndi.service.name" name="internal" description="internal" type="String" default="{jndiName}"/>
```

For more information about OSGi Metatype, see “[Advanced Configuration](#)” on page 1121.

3. **8.5.5.1** Implement the `ResourceFactory` interface if you need Java EE resource reference information. If your service needs Java EE resource reference information, such as `res-auth`, you can register a `ResourceFactory` in the OSGi service registry with the `jndiName` and `creates.objectClass` properties. The `ResourceFactory` service is re-registered automatically with the `osgi.jndi.service.name` property. For example:

```
import com.ibm.wsspi.resource.ResourceFactory;
public class WidgetResourceFactory implements ResourceFactory { ... }

Properties properties = new Properties();
properties.put(ResourceFactory.JNDI_NAME, "widget/abc");
properties.put(ResourceFactory.CREATES_OBJECT_CLASS, Widget.class.getName());
bundleContext.registerService(ResourceFactory.class, new WidgetResourceFactory(), properties);
```

Alternatively, the service could be registered automatically using declarative services and metatype. In that case, you can specify the `creates.objectClass` property as a declarative services property. You do not need to specify the `jndiName` property because it is set automatically from the user configuration with the `<AD id="jndiName">` element in the `metatype.xml` file in step 2, and you do not need an `<AD id="osgi.jndi.service.name">` element in the `metatype.xml` file because the `ResourceFactory` service will be re-registered automatically.

4. Locate an object using the `osgi.jndi.service.name` property with the JNDI name. For example:

```
bundleContext.getServiceReference(DataSource.class, "(osgi.jndi.service.name=jdbc/myds)");
```

Alternatively, you can locate a `ResourceFactory` using the `jndiName` and `creates.objectClass` properties.

5. Update your `metatype.xml` to allow a resource to be specified in `server.xml` using the id of the resource. This allows the resource to be accessed regardless of whether or not the resource has a `jndiName`. For example,

```
<AD id="dataSourceRef" type="String" ibm:type="pid" ibm:reference="com.ibm.ws.jdbc.dataSource" cardinality="1" name="%d
```

If you are using declarative services, you can use an internal attribute to set a `.target` service property with a filter. For example, if your declarative services component has a reference named `dataSource`, you can use the following attribute definition to ensure that the `dataSource` that is referenced by the `dataSourceRef` configuration attribute is used.

```
<AD id="dataSource.target" type="String" default="(service.pid=${dataSourceRef})" ibm:final="true" name="internal" de
```

## Developing a custom TAI as a Liberty feature

You can develop a custom TAI as a Liberty feature by implementing the `com.ibm.wsspi.security.tai.TrustAssociationInterceptor` interface provided in the Liberty server and creating a product extension.

### About this task

For a general view of custom TAI, see “Developing a custom TAI for Liberty” on page 1301.

For more information about product extensions, see “Product extension” on page 577.

**Avoid trouble:** If you have multiple TAIs, you can configure all of them by using either the user feature or the shared library. Do not mix the two TAI configurations.

### Procedure

1. Implement the custom TAI. For more information, see “Developing a custom TAI for Liberty” on page 1301.
2. Convert the implementation class into an OSGi service. You can do the conversion in one of the following ways:
  - Convert your custom TAI class into a Declarative Service (DS) component. For more information, see “Declaring your services to OSGi Declarative Services” on page 1116.
  - Write a new custom TAI class that is a DS component and delegate it to your custom TAI class.
  - Register the custom TAI class directly in the Service Registry (SR) by using the OSGi core APIs. For more information, see “Working with the OSGi service registry” on page 1112.
3. Package the custom TAI as an OSGi bundle and export the custom TAI service. For information on creating an OSGi bundle, see *Creating an OSGi service bundle*.
4. Create a feature manifest to include the OSGi bundle. For more information about feature manifest file, see “Liberty feature manifest files” on page 1097.
5. After the feature is installed into the user product extension location, configure the `server.xml` file with the feature name. For example:

```
<featureManager>
 ...
 <feature>usr:customTaiSample-1.0</feature>
</featureManager>
```

## Dynamic content management

You typically install bundles into the runtime environment by listing them in the `Subsystem-Content` header of the feature manifest file. However, you can also dynamically add and remove OSGi bundles by installing a user-written bundle as part of the `Subsystem-Content` of a user-written feature. The user-written bundle obtains the OSGi bundle context to install and control additional bundles.

### Installing, starting, stopping, and uninstalling bundles in Liberty

**Note:** In the following sections, the user-written feature is called `UserFeatureA` and the user-written bundle is called `FeatureBundleA`.

## Installing bundles

You can write `FeatureBundleA` to obtain the OSGi bundle context, `org.osgi.framework.BundleContext`, by using one of the following methods:

- Implement the `BundleActivator` interface, `org.osgi.framework.BundleActivator`. The `BundleContext` parameter of the `start` method is passed in by the OSGi framework, which is available to the user-written bundle when that bundle is activated. For more information on the `BundleActivator` interface, see “Developing an OSGi bundle with simple activation” on page 1111.
- Implement an available specification, such as OSGi Declarative Services or Blueprint, that provides access to the bundle context through another method or interface. For more information, see “Composing advanced features by using OSGi Declarative Services” on page 1116 and Blueprint bundles.

After `FeatureBundleA` obtains the bundle context, additional bundles can be installed by using either the `installBundle(String location)` or `installBundle(String location, InputStream stream)` methods.

Bundles that are dynamically installed resume state on a default restart. They do not persist across a clean start and require reinstallation. See [Bundle caching](#) for more details.

## Starting bundles

If you want an installed bundle to be started, it is the responsibility of the installing bundle, `FeatureBundleA`, to call the `start` method for the bundle.

## Stopping and uninstalling bundles

If the user-written feature, `UserFeatureA`, is removed from the server configuration, then `FeatureBundleA` is stopped and also uninstalled. Uninstallation of `FeatureBundleA` triggers uninstallation of all the bundles that were installed by `FeatureBundleA`, if they are not already uninstalled. The `org.osgi.framework.Bundle.uninstall()` method is called for each bundle, which stops and uninstalls it. This uninstallation process also applies if `FeatureBundleA` is uninstalled by any other means.

If `UserFeatureA` is removed from the server configuration when the server is stopped, the bundles that were installed by `UserFeatureA` are removed on the next server start. If the start levels of the bundles are unmodified from their defaults, the bundles are removed before they are restarted. If the start levels of the bundles were modified, they might not be removed until after they have restarted.

Other lifecycle management tasks are done by `FeatureBundleA` according to the OSGi core specification, by using the `org.osgi.framework.Bundle` and `org.osgi.framework.BundleContext` interfaces.

## Bundle caching, package visibility, and programming model support in Liberty

### Bundle caching

When the server is shut down, all the currently installed bundles are stopped and OSGi metadata is persisted to a bundle cache. On a default start, these installed bundles are returned to their previous state. On a clean start, any bundles that were installed by `FeatureBundleA` have their persistent data deleted. As a result, on a clean start these bundles are not resumed.

`FeatureBundleA` itself is resumed because it is reinstalled by the feature manager, provided `UserFeatureA` is still in the server configuration. If you want to reinstall any bundles after a clean start, it is the responsibility of `FeatureBundleA` to do the reinstallation. You are not notified about a clean start, but you can check whether a bundle is installed by using the OSGi `BundleContext` `getBundle(String location)` method.

### Package visibility

Bundles that are dynamically installed, and are not listed in the `Subsystem-Content` header of the feature manifest file, have the following visibility:

- Dynamically installed bundles can import any API and SPI packages provided by the currently configured set of features.

- Packages that are exported by other bundles within the same product extension that are not declared as API or SPI, are not visible to dynamically installed bundles.
- Packages that are exported from dynamically installed bundles cannot be declared as API or SPI.
- There are no restrictions on importing packages that are exported from dynamically installed bundles.

### Programming model support

Dynamically installed bundles can use implementations of OSGi enterprise specifications provided that the appropriate runtime features are configured to enable them.

---

## Packaging and installing Liberty features

You can package and install Liberty features to WebSphere Application Server Liberty.

### About this task

A Liberty feature may be packaged as a subsystem archive, as defined by the OSGi Enterprise specification (5.0). A subsystem archive is a compressed file with an `.esa` extension that includes the feature manifest and the resource files that constitute the feature. The WebSphere Application Server developer tools will import and export Liberty features by using the subsystem archive format, and features can be installed to a Liberty runtime environment by using the **featureManager** command if they are in the `.esa` format.

### Procedure

- You can manually install a Liberty features onto Liberty.
  - To install the feature to the Liberty kernel, the feature manifest file must be in the `${wlp.install.dir}/lib/features` directory and related bundles in the `${wlp.install.dir}/lib` directory;
  - To install the feature into the user configuration, the feature manifest file must be in the `${wlp.user.dir}/extension/lib/features` directory, and related bundles must be in the `${wlp.user.dir}/extension/lib` directory;
  - To install the feature into a product extension, the feature manifest file and related bundles must be in the product extension directory. The product extension is registered in the `${wlp.user.dir}/etc/extension/lib/features` directory using a `<extension-name>.properties` file. For more information, see “Product extension” on page 577.
- You can also use the **featureManager** command that is provided in the `${wlp.install.dir}/bin` directory to install a Liberty feature if it is packaged as an `.esa` file.
  - To install a feature as a user feature, use the following command:
 

```
featureManager install my_feature.esa --to=usr
```
  - To install a feature to a product extension location, use the following command:
 

```
featureManager install my_feature.esa --to=my_extension
```

For more information about the **featureManager** command, see “featureManager command” on page 865.

## Provide product information for your feature extension

You can provide Version Product information for your Feature Extension

You can provide Version Product information for your Feature Extensions by providing the installation with a uniquely named product information properties file in the `lib/versions` directory of the extension. The file extension must be `.properties`.

You can add Version Product information manually or by using the featureManager script's installation option under the following conditions:

- The subsystem manifest file in the feature archive contains a subsystem-content entry that points to the product information properties file that is being installed.
- The product information properties file can be installed only if the feature that is being installed defines the **IBM-Feature-Version**.
- The product information properties file in the feature archive (.esa) has the following path `wlp/lib/versions/filename.properties`.

The following properties can be specified in the product information properties file:

```
com.ibm.websphere.productId=yourProductID
com.ibm.websphere.productOwner=TheProductOwner
com.ibm.websphere.productVersion=yourProductVersion
com.ibm.websphere.productName=yourProductName
com.ibm.websphere.productInstallType=yourProductInstallType
com.ibm.websphere.productEdition=yourProductEdition
com.ibm.websphere.productQualifier=yourProductQualifier
```

If you want to override a particular feature extension information, include the following property in your product information properties file:

```
com.ibm.websphere.productReplaces=theProductIdToReplace
```

If your product is eligible for Getting Started Subcapacity Pricing (GSSP), set the following property to true in your product information properties file:

```
com.ibm.websphere.gssp=true
```

The following is an example of how to define subsystem entry content for installing the feature extension by using the featureManager script, `subsystem.mf`:

```
Subsystem-ManifestVersion:1
...
Subsystem-Content: userProdExt; version="[1,1.0.100]",
user.ext.version.info; type="file"; location="lib/versions/user.ext.version.info.properties"
IBM-Feature-Version: 2
```

The following is an example that shows the output of the `productInfo` script using the `version` option that shows the product name and version entries that are specified in the product information properties file:

```
com.ibm.websphere.productId=XYZ Product ID
com.ibm.websphere.productOwner=XYZ Inc
com.ibm.websphere.productVersion=1.0.0
com.ibm.websphere.productName=XYZ User Product
com.ibm.websphere.productInstallType=Archive
com.ibm.websphere.productEdition=Enterprise Edition
```

```
Command:
productInfo version
```

```
Output:
Product name: Websphere Application Server
Product version: 8.5.5.0
```

```
Product name: XYZ User Product
Product version: 1.0.0
```

---

## Embedding Liberty in your applications

You can use the System Programming Interfaces (SPIs) that are provided by Liberty to configure, control, and monitor a Liberty server in your applications.



## About this task

Liberty provides the following SPIs to start or stop a Liberty server:

- `com.ibm.wsspi.kernel.embeddable.Server`
- `com.ibm.wsspi.kernel.embeddable.ServerBuilder`

Use a Future object to store the result of a start or stop operation. The return codes that are used by embedded operations are the same as the return codes used by the **server** command. For more information about return codes, JVM options used by the server script, and the process environment used by the server script, see “server command options” on page 950.

Additionally, you can receive asynchronous notifications when the server is starting, has started, or has stopped by creating your own class that implements the `com.ibm.wsspi.kernel.embeddable.ServerEventListener` interface.

**Note:** To create an instance of an embedded server within your application, you must carry out the following steps:

- Include the `ws-server.jar` file on the class path. The `ws-server.jar` file is in the `${wlp.install.dir}/bin/tools` directory of the Liberty installation.
- Specify the name of the target server. The target server must exist.
- Optional: Configure the `ws-javaagent.jar` file with the `-javaagent` JVM option. The `ws-javaagent.jar` file is in the `${wlp.install.dir}/bin/tools` directory of the Liberty installation. You are advised to configure the `ws-javaagent.jar` file, but it is not mandatory unless you use capabilities of the server that require it, such as monitoring or trace. If you contact IBM support, you might need to provide trace, and if so, you must start the server with the `ws-javaagent.jar` file, even if you do not normally use it.

**Note:** In an embedded environment:

- Environment variables are not checked, and the `jvm.options` and `server.env` files are not read.
- Management of the JVM and environment is assumed to be managed by the caller.

## Procedure

1. Import the SPIs into your caller class and define the arguments that are required to operate the Liberty server.

```
import com.ibm.wsspi.kernel.embeddable.Server;
import com.ibm.wsspi.kernel.embeddable.ServerBuilder;

public class MyEmbeddedServer {
 String serverName="defaultServer";
 File userDir = new File("usr");
 File outputDir = new File("usr/servers/");
 ...
}
```

Where

- The **serverName** is required, and must match the name of a previously created server.
  - The **userDir** is optional and used to set the path of the user directory. By default, the user directory is `${wlp.user.dir}`.
  - The **outputDir** is optional and used to set the path of the output directory. By default, the output directory is `${wlp.user.dir}/servers`.
2. Initialize the server by using the `ServerBuilder` class.

```

ServerBuilder sb = new ServerBuilder();
Server libertyServer = sb.setName(serverName)
 .setUserDir(userDir)
 .setOutputDir(outputDir)
 .build();

```

3. Call the `Server.start()` method to start the server. Call `get()` on the future to block until the start operation completes. Use one of the following to determine whether the server started successfully:

- Check the returned result code.
- Use the `successful()` method.
- If the server is started, the `server.isRunning()` method returns true.

```

Future<Result> startReturnCode = libertyServer.start();
Result result = startReturnCode.get(); // block until operation complete, if necessary
System.out.println("Start returned: success=" + result.successful() + ", rc=" + result.getReturnCode() + ", ex=" + result.getException());

```

4. Call the `Server.stop()` method to stop the server. Call `get()` on the future to block until the stop operation completes. Use one of the following to determine whether the server stopped successfully:

- Check the returned result code.
- Use the `successful()` method.
- If the server is stopped, the `server.isRunning()` method returns false.

```

Future<Result> stopReturnCode = libertyServer.stop();
Result result = stopReturnCode.get(); // block until operation complete, if necessary
System.out.println("Stop returned: success=" + result.successful() + ", rc=" + result.getReturnCode() + ", ex=" + result.getException());

```

5. Implement the `ServerEventListener` interface. If you implement the `ServerEventListener` interface, you can receive notifications when the server is started or stopped.

```

// update the class declaration to indicate that it implements ServerEventListener
public class MyEmbeddedServer implements ServerEventListener {
 ...
 MyEmbeddedServer() throws ServerException {
 // set the listener via the server builder
 ServerBuilder sb = new ServerBuilder();
 Server libertyServer = sb.setName(serverName)
 .setServerEventListener(this)
 .build();
 }

 ...
 @Override
 public void serverEvent(ServerEvent event) {
 // provide an implementation of the serverEvent method
 System.out.println("serverEvent: " + event);
 }
}

```

---

## Creating Liberty servers from custom configurations

You can create a server from a custom configuration for any environment that you require.

### About this task

The Liberty server script create command offers a `--template` option. You can use this option to support server creation from a custom configuration located within a `templates/servers/<template-name>` sub-directory of your product extension. Custom server templates must contain at least a `server.xml` file, and can contain any configuration files, for example: `bootstrap.properties` or `jvm.options`.

### Procedure

You can use the `--template` option in the following way: `server create --template=<extension-name>:<template-name>`

---

## Chapter 7. Securing Liberty and its applications

This information applies to all types of applications that are deployed on Liberty.

### About this task

Security in Liberty supports all the Servlet 3.0 security features and secured Java JMX connections. The following Liberty features are applicable to security in Liberty:

- `appSecurity-2.0` enables security for web applications when the `servlet-3.0` feature is present and for EJB components when the `ejbLite-3.1` feature is present.
- `ssl-1.0` enables SSL connections using HTTPS.
- `restConnector-1.0` enables remote access by JMX client through a REST-based connector.
- `oauth-2.0` enables authorization to resources by using the OAuth 2.0 protocol.
- `ldapRegistry-3.0` provides support for the LDAP user registry.

To learn about how security works in Liberty, see “Security” on page 581.

**Best practice:** There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty. If you see any differences in the configuration created by the developer tools and the examples, modify the configuration to fit the configuration in the examples for that feature.

### Procedure

- Use `quickStartSecurity` for minimal security configuration
- Secure communications in Liberty
- Access a secured JMX connector in Liberty
- Authenticate users in Liberty
- Authorize access to resources in Liberty
- Secure a database access application
- Develop extensions to the Liberty security infrastructure

---

## Getting started with security in Liberty


You can use the `<quickStartSecurity>` element to quickly enable a simple (one user) security setup for Liberty.

### About this task

You can set up a secured Liberty server and web application by following some basic configuration steps. Configuration actions within Liberty are dynamic, which means the configuration updates take effect without having to restart the server.

### Procedure

1. Create and start your server.

-  On Windows systems:  

```
server.bat create MyNewServer
server.bat start MyNewServer
```

- **AIX** **Linux** **UNIX** **HP-UX** **Solaris** **IBM i** On all systems other than Windows systems:

```
server create MyNewServer
server start MyNewServer
```

2. Include the appSecurity-2.0 and servlet-3.0 features in the server.xml file.

The server.xml file is in the server directory of *myNewServer*, for example, `wlp\usr\servers\myNewServer\server.xml`.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>servlet-3.0</feature>
</featureManager>
```

3. Define the user name and password that is to be granted the Administrator role for server management activities.

```
<quickStartSecurity userName="Bob" userPassword="bobpwd" />
```

**Note:** Choose a user name and password that are meaningful to you. Never use the name and password in the example for your applications.

4. Configure the deployment descriptor with relevant security constraints to protect web resource. For example, use `<auth-constraint>` and `<role-name>` elements to define a role that can access web resource.

The following example web.xml file shows that access to all the URIs in the application is protected by the testing role.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app id="myWebApp">

 <!-- SERVLET DEFINITIONS -->
 <servlet id="Default">
 <servlet-name>myWebApp</servlet-name>
 <servlet-class>com.web.app.MyWebAppServlet</servlet-class>
 <load-on-startup/>
 </servlet>

 <!-- SERVLET MAPPINGS -->
 <servlet-mapping id="ServletMapping_Default">
 <servlet-name>myWebApp</servlet-name>
 <url-pattern>/*</url-pattern>
 </servlet-mapping>

 <!-- SECURITY ROLES -->
 <security-role>
 <role-name>testing</role-name>
 </security-role>

 <!-- SECURITY CONSTRAINTS -->
 <security-constraint>
 <web-resource-collection>
 <url-pattern>/*</url-pattern>
 </web-resource-collection>
 <auth-constraint>
 <role-name>testing</role-name>
 </auth-constraint>
 </security-constraint>

 <!-- AUTHENTICATION METHOD: Basic authentication -->
 <login-config>
```

```
<auth-method>BASIC</auth-method>
</login-config>
```

```
</web-app>
```

5. Configure your application in the `server.xml` file.

In the following example, the user Bob is mapped to the testing role of the application:

```
<application type="war" id="myWebApp" name="myWebApp"
 location="${server.config.dir}/apps/myWebApp.war">
 <application-bnd>
 <security-role name="testing">
 <user name="Bob" />
 </security-role>
 </application-bnd>
</application>
```

6. Access your application and log in with the user name Bob. The default URL for the myWebApp application is `http://localhost:9080/myWebApp`

## Results

You have now secured your application.

## Quick overview of security

To understand the basic workflow of security in Liberty, some common security terms are detailed along with an example.

### Security key terms

#### Authentication

Authentication confirms the identity of a user. The most common form of authentication is user name and password, such as through basic authentication or form login for web applications. When a user is authenticated, the source of a request is represented as a Subject object at run time.

#### Authorization

Authorization determines whether a user has access to a given role within the system. The Java EE model uses subjects, roles, and role mappings to determine if access is allowed.

**Role** A role is defined within the Java EE application. Some roles, such as the Administrator role, are predefined by the system. Other roles are defined by the application developer. In Java EE, subjects are usually granted or denied access to a role based on the roles they perform within the application.

#### Subject

A subject is both a general term and a Java object: `javax.security.auth.Subject`. Generally, the term subject means active entities within the system, such as users on the system, and even the system process itself.

### Security workflow example

The following example demonstrates how the security is applied when a user requests access to a resource. For example, a user Bob wants to access a servlet myWebApp. See the code samples in “Getting started with security in Liberty” on page 1147.

To access the servlet myWebApp, the following conditions must be true:

1. Bob must be able to log in to the system because the servlet is protected.
2. Bob must be in the testing role because the servlet is restricted by using an `auth-constraint` element in the deployment descriptor.

If Bob cannot log in to the system, or Bob is not in the testing role, then the access to the servlet myWebApp is denied.

Another user Alice can log in to the system because Alice is a valid user. But Alice is not in the testing role. An HTTP 403 error (Access Denied/Forbidden) displays when Alice logs in.

## Setting up BasicRegistry and role mapping on Liberty

You can configure Liberty to authenticate and authorize users by using a basic user registry.

### Before you begin

The Liberty features appSecurity-2.0 and servlet-3.0 must be enabled in the server.xml file.

For more information about security configuration in Liberty, see “Getting started with security in Liberty” on page 1147.

### About this task

You can set up a basic user registry and configure more role mapping in the server.xml file for a Liberty server by going through the following steps.

### Procedure

1. Configure the basic registry as follows. Use a user name and password that are meaningful to you. Never use the name and password from this example in your applications.

```
<basicRegistry id="basic" realm="WebRealm">
 <user name="Bob" password="bobpwd" />
</basicRegistry>
```

2. Optional: Grant the user or group the Administrator role if the user, or group of users, is used to perform remote system management activities. This step is done automatically when using the quickStartSecurity element or may be accomplished by adding the administrator-role element to the server.xml file as shown.

```
<administrator-role>
 <user>Bob</user>
 <group>myAdmins</group>
</administrator-role>
```

3. Encode the password within the configuration. You can get the encoded value by using the securityUtility encode task.
4. Optional: Add additional users. Make sure that each user name is unique.

```
<basicRegistry id="basic" realm="WebRealm">
 <user name="Bob" password="bobpwd" />
 <user name="user1" password="user1pwd" />
 <user name="user2" password="user2pwd" />
</basicRegistry>
```

5. Create groups for users. Make sure that each group name must be unique.

```
<basicRegistry id="basic" realm="WebRealm">
 <user name="Bob" password="bobpwd" />
 <user name="user1" password="user1pwd" />
 <user name="user2" password="user2pwd" />

 <group name="myAdmins">
 <member name="Bob" />
 <member name="user1" />
 </group>

 <group name="users">
```

```

 <member name="user1" />
 <member name="user2" />
 </group>
</basicRegistry>

```

6. Assign some users and groups to the testing role of an application.

```

<application type="war" id="myWebApp" name="myWebApp"
 location="{server.config.dir}/apps/myWebApp.war">
 <application-bnd>
 <security-role name="testing">
 <user name="Bob" />
 <user name="user1" />
 <group name="users" />
 </security-role>
 </application-bnd>
</application>

```

## What to do next

Configure security-related elements in the deployment descriptor of your application. See “Getting started with security in Liberty” on page 1147 for a sample `web.xml` file.

---

## Securing communications in Liberty

You can configure the Liberty server to provide secure communications between a client and the server.

### About this task

Communications are secured with Secure Sockets Layer (SSL) protocol. The SSL protocol provides transport layer security including authenticity, data signing, and data encryption to ensure a secure connection between a client and server that uses WebSphere® Application Server. The foundation technology for SSL is public key cryptography, which guarantees that when an entity encrypts data using its public key, only entities with the corresponding private key can decrypt that data. The Liberty Server uses Java Secure Sockets Extension (JSSE) as the SSL implementation for secure connections. JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure that secure connectivity exists across most protocols. JSSE relies on X.509 certificate-based asymmetric key pairs for secure connection protection and some data encryption. Key pairs effectively encrypt session-based secret keys that encrypt larger blocks of data. The SSL implementation manages the X.509 certificates.

To configure secure communications, you can either specify a minimal SSL configuration or a detailed SSL configuration in the `server.xml` file. The minimal configuration only requires the SSL feature and a keystore entry to be specified. There are several security configuration examples on the [WASdev.net](http://WASdev.net) website for reference when configuring security for your applications on Liberty.

The SSL configuration that is designated as the default SSL configuration is used to create the process's default `SSLContext` using the `SSLContext.setDefault()` method. The default SSL configuration can be the minimal SSL configuration, or the configuration that is identified by the `sslRef` attribute on the `sslDefault` element if multiple SSL configurations are defined. Because the default `SSLContext` is set on the process, the `javax.net.ssl.keyStore` and `javax.net.ssl.trustStore` properties will not be recognized.

### Procedure

- Enable SSL communications between a client and a Liberty server
- Optional: Create a keystore from the command prompt
- Optional: Encode passwords from the command prompt
- Optional: Configure client certificate authentication between your application and the Liberty server

## Enabling SSL communication in Liberty

To enable SSL communication in Liberty, there is a minimal set of SSL configuration options. It assumes most of the SSL options require some keystore configuration information.

### About this task

SSL client authentication occurs during the connection handshake by using SSL certificates. The SSL handshake is a series of messages that are exchanged over the SSL protocol to negotiate for connection-specific protection. During the handshake, the secure server requests that the client send back a certificate or certificate chain for the authentication. To enable SSL in Liberty, you add the `ssl-1.0` Liberty feature to the configuration root document file, `server.xml`, along with code of the keystore information for authentication.

By default, the path and file name for the configuration root document file is `path_to_liberty/wlp/usr/servers/server_name/server.xml`. `path_to_liberty` is the location that you installed Liberty on your operating system, and `server_name` is the name of your server. However, you can change the path. See “Customizing the Liberty environment” on page 947.

### Procedure

1. Enable the `ssl-1.0` Liberty feature in the `server.xml` file.

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>
```

**Note:** If application security is required and security information is redirected to a secure port, you must add the `appSecurity-2.0` Liberty feature to the `server.xml` file.

2. Add the keystore service object entry to the `server.xml` file. The `keyStore` element is called `defaultKeyStore` and contains the keystore password. The password can be entered in clear text or encoded. The `securityUtility encode` option can be used to encode the password.

```
<keyStore id="defaultKeyStore" password="yourPassword" />
```

This configuration is the minimum that is needed to create an SSL configuration. In this configuration, the server creates the keystore and certificate if it does not exist during SSL initialization. The password that is provided must be at least 6 characters long. The keystore is assumed to be a JKS keystore that is called `key.jks` in the server `home/resources/security` directory. If the file does not exist the server creates it for you. If the server creates the keystore file, it also creates the certificate inside of it. The certificate is a self-signed certificate with a validity period of 365 days, the CN value of the certificate's subjectDN is the host name of the machine where the server is running, and has a signature algorithm of `SHA256withRSA`.

**Note:** When the use of a collective controller is not practical, perhaps there is only one or two Liberty servers, a self-signed certificate can be used to restrict the number of clients that can connect to the Liberty member server. It is suggested that an IHS server is used in front of the Liberty servers, where an appropriate CA signed certificate can be used, along with CN whitelisting to control which clients can connect to HIS. A trusted channel between IHS and the Liberty member server can be maintained by using the self signed certificate.

An example of a SAF keyring in the minimal configuration:

```
<keyStore id="defaultKeyStore" location="safkeyring:///WASKeyring"
 type="JCERACFKS" password="password" fileBased="false"
 readOnly="true" />
```

RACF<sup>®</sup> keyring needs to be set up before you configure them for use by the Liberty server. The server does not create certificates and add them to RACF.

The single keystore entry for a minimal SSL configuration can be extended to include the location and type as well.

```
<keyStore id="defaultKeyStore" location="myKeyStore.p12" password="yourPassword" type="PKCS12"/>
```



The **location** parameter can be an absolute path to the keystore file. If it is an absolute path, then the keystore file is assumed to have been already created. Keystore of other types can also be specified in the minimal SSL configuration if the keystore file is already created. When the minimal SSL configuration is used, the SSL configuration defaults are used to create the SSL context for an SSL handshake. The configuration protocol is `SSL_TLS` by default. The HIGH ciphers, 128 bit, and higher cipher suites can be used.

## SSL configuration attributes

SSL configurations contain attributes that you use to control the behavior of the server SSL transport layer on Liberty. This topic iterates all the settings available for an SSL configuration.

## SSL Feature

To enable SSL on a server, the SSL feature must be included in the `server.xml` file:

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>
```

## SSL Default

You can have multiple SSL configurations configured. If more than one SSL configuration is configured, then the default SSL configuration must be specified in the `server.xml` file that uses the `sslDefault` service configuration.

*Table 84. Attribute of the `sslDefault` element.* This table describes the attribute of the `sslDefault` element.

Attribute	Description	Default Value
<b>sslRef</b>	The <code>sslRef</code> attribute specifies the name of the SSL configuration to be used as the default.	The default SSL Configuration name is <code>defaultSSLConfig</code> .

In the `server.xml` file, the entry is as follows:

```
<sslDefault sslRef="mySSLSettings" />
```

## SSL Configuration

You use the SSL configuration attributes to customize the SSL environment to suit your needs. These attributes can be set on the `ssl` service configuration element in the `server.xml` file.

*Table 85. Attributes of the `SSL` element.* This table describes the attributes of the `ssl` element.

Attribute	Description	Default Value
<b>id</b>	The <code>id</code> attribute assigns a unique name to the SSL configuration object.	No default value; a unique name must be specified.
<b>keyStoreRef</b>	The <code>keyStoreRef</code> attribute names the keystore service object that defines the SSL configurations keystore. The keystore holds the key that is required to make an SSL connection.	No default value; a keystore reference must be specified.
<b>trustStoreRef</b>	The <code>trustStoreRef</code> attribute names the keystore service object that defines the SSL configurations truststore. The truststore holds certificates that are required for signing verification.	<code>trustStoreRef</code> is an optional attribute. If the reference is missing, the keystore that is specified by <code>keyStoreRef</code> is used.

Table 85. Attributes of the SSL element (continued). This table describes the attributes of the `ssl` element.

Attribute	Description	Default Value
<b>clientAuthentication</b>	The <b>clientAuthentication</b> attribute determines whether SSL client authentication is required.	Default value is <b>false</b> .
<b>clientAuthenticationSupported</b>	The <b>clientAuthenticationSupported</b> attribute determines whether SSL client authentication is supported. The client does not have to supply a client certificate. If the <b>clientAuthentication</b> attribute is set to true, the value of the <b>clientAuthenticationSupported</b> attribute is overwritten.	Default value is <b>false</b> .
<b>sslProtocol</b>	The <b>sslProtocol</b> attribute defines the SSL handshake protocol. The protocol can be SDK-dependent, so if you modify the protocol make sure that the value is supported by the SDK you are running under.	Default value is <b>SSL_TLS</b> .
<b>securityLevel</b>	<p>The <b>securityLevel</b> attribute determines the cipher suite group to be used by the SSL handshake. The attribute has one of the following values:</p> <ul style="list-style-type: none"> <li>• <b>HIGH</b> (128-bit ciphers and higher)</li> <li>• <b>MEDIUM</b> (40-bit ciphers)</li> <li>• <b>WEAK</b> (for all ciphers without encryption)</li> <li>• <b>CUSTOM</b> (if the cipher suite group is customized).</li> </ul> <p>When you set the <b>enabledCiphers</b> attribute with a specific list of ciphers, the system ignores this attribute.</p>	Default value is <b>HIGH</b> .
<b>enabledCiphers</b>	The <b>enabledCiphers</b> attribute is used to specify a unique list of cipher suites. Separate each cipher suite in the list with a space. If the <b>enabledCiphers</b> attribute is set, then the <b>securityLevel</b> attribute is ignored.	No default value.
<b>serverKeyAlias</b>	The <b>serverKeyAlias</b> attribute names the key in the keystore to be used as the SSL configurations key. This attribute is only required if the keystore has more than one key entry in it. If the keystore has more than one key entry and this attribute does not specify a key, then the JSSE picks a key.	No default value.
<b>clientKeyAlias</b>	The <b>clientKeyAlias</b> attribute names the key in the keystore to be used as the key for SSL configuration when <b>clientAuthentication</b> is enabled. The attribute is only required if the keystore contains more than 1 key entry.	No default value.

Note:

- The key manager is used by the SSL handshake to determine what certificate alias to use. The key manager is not configured in the `server.xml` file. It is retrieved from the security property `ssl.KeyManagerFactory.algorithm` of the SDK.
- The trust manager is used by the SSL handshake to make trust decisions. The trust manager is not configured in the `server.xml` file. It is retrieved from the security property `ssl.TrustManagerFactory.algorithm` of the SDK.

Here is an example of how the `ssl` element is configured in the `server.xml` file:

```
<!-- Simple ssl configuration service object. This assumes there is a keystore object named -->
<!-- defaultKeyStore and a truststore object named defaultTrustStore in the server.xml file. -->
 <ssl id="myDefaultSSLConfig"
 keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" />
```

```
<!-- A ssl configuration service object that enabled clientAuthentication -->
<!-- and specifies the TLS protocol be used. -->
 <ssl id="myDefaultSSLConfig"
 keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore"
 clientAuthentication="true"
 sslProtocol="TLS" />
```

```
<!-- An SSL configuration service object that names the serverKeyAlias -->
<!-- to be used by the handshake. This assumes there is a certificate -->
<!-- called "default" in the keystore defined by keyStoreRef. -->
 <ssl id="myDefaultSSLConfig"
 keyStoreRef="defaultKeyStore"
 serverKeyAlias="default" />
```

## Keystore Configuration

The keystore configuration consists of the attributes that are required to load a keystore. These attributes can be set on the keystore service configuration in the `server.xml` file.

*Table 86. Attributes of the keystore element.* This table explains the attributes of the keystore element.

Attribute	Description	Default Value
<b>id</b>	The <b>id</b> attribute defines a unique identifier of the keystore object.	No default value, a unique name must be specified.
<b>location</b>	The <b>location</b> attribute specifies the keystore file name. The value can include the absolute path to the file. If the absolute path is not provided, then the code looks for the file in the <code>\${server.config.dir}/resources/security</code> directory.	In the SSL minimal configuration, the location of the file is assumed to be <code>\${server.config.dir}/resources/security/key.jks</code> .
<b>type</b>	The <b>type</b> attribute specifies the type of the keystore. Check that the keystore type that you specify is supported by the SDK you are running on.	Default value is <b>jks</b> .

Table 86. Attributes of the keystore element (continued). This table explains the attributes of the keystore element.

Attribute	Description	Default Value
<b>password</b>	The <b>password</b> attribute specifies the password that is used to load the keystore file. The password can be stored either in clear text or encoded. For information about how to encode the password, see the securityUtility encode option.	Must be provided.
<b>provider</b>	The <b>provider</b> attribute specifies the provider to be used to load the keystore. Some keystore types required a provider other than the SDK default.	By default no provider is specified.
<b>fileBased</b>	The <b>fileBased</b> attribute specifies whether the keystore is file-based.	Default value is <b>true</b> .
<b>pollingRate</b>	The rate at which the server checks for updates to a keystore file.	<b>500ms</b> .
<b>updateTrigger</b>	The method that is used to trigger the server to reload a keystore file. Specify <b>polled</b> to enable the server for checking the keystore file for changes, <b>mbean</b> to enable the server to wait for an mbean to reload the keystore file, or <b>disabled</b> to disable file monitoring.	<b>disabled</b> .

Keystore files can be reloaded by the server if the **updateTrigger** attribute is set to **polled** or **mbean**. If **polled** is enabled, then the server monitors the keystore file for changes based on the rate set in the **pollingRate** attribute. If the **updateTrigger** attribute is set to **mbean** then the server will reload the keystore file when it receives notification from the `WebSphere:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean` MBean. File monitoring is disabled by default.

Here is an example of how the keystore element is configured in the `server.xml` file:

```
<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be monitored every 5 seconds -->
<!-- for updates -->
 <keyStore id="defaultKeyStore"
 location="MyKeyStoreFile.jks"
 type="JKS" password="myPassword"
 pollingRate="5s"
 updateTrigger="polled" />

<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be reloaded when the server -->
<!-- recieves an mbean notification to do so -->
 <keyStore id="defaultKeyStore"
 location="MyKeyStoreFile.jks"
 type="JKS" password="myPassword"
 updateTrigger="mbean" />
```

## Full SSL Configuration Example

Here is an example of a full SSL configuration in the `server.xml` file. This example has the following SSL configurations:

- `defaultSSLSettings`
- `mySSLSettings`

By default, the SSL configuration is set to `defaultSSLConfig`.

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>

<!-- default SSL configuration is defaultSSLSettings -->
<sslDefault sslRef="defaultSSLSettings" />
<ssl id="defaultSSLSettings"
 keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore"
 clientAuthenticationSupported="true" />
<keyStore id="defaultKeyStore"
 location="key.jks"
 type="JKS" password="defaultPWD" />
<keyStore id="defaultTrustStore"
 location="trust.jks"
 type="JKS" password="defaultPWD" />

<ssl id="mySSLSettings"
 keyStoreRef="myKeyStore"
 trustStoreRef="myTrustStore"
 clientAuthentication="true" />
<keyStore id="LDAPKeyStore"
 location="{server.config.dir}/myKey.p12"
 type="PKCS12"
 password="{xor}CDo9Hgw=" />
<keyStore id="LDAPTrustStore"
 location="{server.config.dir}/myTrust.p12"
 type="PKCS12"
 password="{xor}CDo9Hgw=" />
```

## Keystores

Liberty can create only a keystore type of Java Keystore (JKS). Support for other types of keystore in Liberty can depend on what is supported by the underlying Java Runtime Environment (JRE). The following are the different keystore types in Liberty.

For more information on configuration attributes of the keystore element, see “SSL configuration attributes” on page 1153.

## JKS and JCEKS

Java Keystore (JKS) and Java Cryptography Extensions Keystore (JCEKS) are common between the IBM JRE and the Oracle JRE, and can be configured the same using either JRE. JKS is the default keystore type in Liberty, and the only type of keystore Liberty can create. If no keystore type is specified in the configuration, JKS is used.

An example of JKS keystore configuration is as follows:

```
<keyStore id="sampleJKSKeyStore"
 location="MyKeyStoreFile.jks"
 type="JKS" password="myPassword" />
```

An example of JCEKS keystore configuration is as follows:

```
<keyStore id="sampleJCEKSKeyStore"
 location="MyKeyStoreFile.jceks"
 type="JCEKS" password="myPassword" />
```

8.5.5.9

## PKCS11 keystore

A hardware cryptographic keystore can be configured so that the Liberty server can be used to provide cryptographic token support.

The user must provide a hardware device-specific configuration file. The configuration file is a text file that contains entries in the format of attribute = value. The file must contain at least the name and library attribute. For example:

```
name = HWDevice
library = /opt/foo/lib/libpkcs11.so
```

The name attribute is a name that is being given to this instance of the device. The library attribute contains a path to the library provided by the hardware device to access the device. The configuration file can also contain configuration data specific to the hardware device.

To configure a PKCS11 keystore in Liberty the keystore element must contain the following fields:

- id - Uniquely identify the keystore element in the configuration.
- location - The path to the hardware device-specific configuration file.
- type - PKCS11 must be specified as the keystore type.
- fileBased - Must be false to identify this keystore as a device.
- password - Password that is needed to access keys in the device.
- provider - The provider that is needed. For the IBM JRE, the value must be IBMPKCS11Impl and for Oracle JRE it must be SunPKCS11.

Here is an example configuration:

```
<keyStore id="hwKeyStore"
 location="${server.config.dir}/HWCrypto.cfg"
 type="PKCS11"
 fileBased="false"
 password="{xor}Lz4sLCgwLTs="
 provider="IBMPKCS11Impl"/>
```

## PKCS12 keystore

Public Key Cryptography Standards #12 (PKCS12) keystore can be used, but not created by Liberty, when you use the IBM JRE. An example of PKCS12 keystore configuration is as follows:

```
<keyStore id="samplePKCS12KeyStore"
 location="MyKeyStoreFile.p12"
 type="PKCS12" password="myPassword" />
```

## CMS keystore

CMS keystore can be configured, but not created by Liberty, when you use the IBM JRE. However, some special configuration is required. The CMS provider is not available by default on the IBM JRE, therefore it must be added to the provider list in the java.security file of the IBM JRE. In the following example, the com.ibm.security.cmskeystore.CMSProvider class is added to the end of the list. Ensure that the provider number is correct in the provider list. Liberty does not use the CMS keystore stash file to gain access to the keystore.

```

security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.security.jgss.IBMJGSSProvider
security.provider.4=com.ibm.security.cert.IBMCertPath
security.provider.5=com.ibm.security.sasl.IBMSASL
security.provider.6=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.7=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.8=org.apache.harmony.security.provider.PolicyProvider
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNego
security.provider.10=com.ibm.security.cmskeystore.CMSProvider

```

To use the CMS keystore, the configuration in the server.xml file is as follows:

```

<keyStore id="sampleCMSKeyStore"
password="myPassword"
location="MyKeyStoreFile.kdb"
provider="IBMCMSProvider"
type="CMSKS"/>

```

## Enabling the IBM JCE Hybrid Provider for Liberty

The IBM JCE Hybrid Provider IBMJCEHYBRID, is for use by an application that is designed to use cryptographic hardware and processors when they are available, but continues without those cryptographic features when they are not available. Using the IBMJCEHYBRID provider enables an application to take advantage of JCE providers without having to include complex error handling for when cryptographic features are not available.

### Before you begin

Ensure that the IBMJDK running on z/OS is at Java 7 SR3 or later.

### About this task

The IBMJCEHYBRID provider does not do any cryptographic operations, but routes requests to JCE providers registered with the Java Security Framework. The IBMJCEHYBRID provider must be the first JCE provider in the active JVM provider list, which is initialized from the java.security.provider list. The IBMJCEHYBRID provider routes requests to, and provides failover for, JCE providers according to the security provider registrations done at JVM initialization. This function enables an application to take advantage of cryptographic features when they are available and to use a provider that does not depend on these features when they are not available.

### Procedure

1. Add the provider to the java.security file with the hardware cryptographic provider.

```

security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
security.provider.2=com.ibm.crypto.ibmjcehybrid.provider.IBMJCEHYBRID
security.provider.3=com.ibm.crypto.hdwrcCA.provider.IBMJCECCA
security.provider.4=com.ibm.crypto.provider.IBMJCE
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.sasl.IBMSASL

```

2. Configure the keyring in file server.xml to set the location to use safkeyringhybrid, and the type to JCEHYBRIDRACFKS. The following example shows the definition of a minimal SSL configuration keystore.

```

<keyStore id="defaultKeyStore" location="safkeyringhybrid:///mykeyring" type="JCEHYBRIDRACFKS"
password="{xor}Lz4sLCgwTs=" fileBased="false" readOnly="true"/>

```

## SSL defaults in Liberty

8.5.5.7

Specifies the default SSL certificate, keystore, and configuration in Liberty.

## Default Certificate and keystore

As a convenience tool to help developers get up and running, users can use the **createSSLCertificate** parameter in the **securityUtility** command to create self-signed certificates. Users can either call the tool directly from the command line or let the server call it to create the default certificate and keystore on server startup.

The server creates the default keystore and certificate if a user has a keystore element that is called `defaultKeyStore` in the `server.xml` file. For example:

```
<keyStore id="defaultKeyStore" password="yourPassword" />
```

If a keystore configuration for the `defaultKeyStore` is in place and the keystore does not exist when the server starts, the server calls the **createSSLCertificate** parameter to create the keystore with a password that is specified in the configuration.

Default keystore details:

- Location: The keystore file is called `key.jks` and is in the `server` or `clients resources/security` directory.
- Keystore type: The keystore type is JKS.
- Password: Password that is provided in the configuration.

Default certificate that is created by Liberty details:

- Type: The certificate is a self-signed certificate.
- Size: The default certificate size is 2048.
- Signature algorithm: The signature algorithm for the certificate is SHA256WITHRSA.
- Validity: The certificate is valid for 365 days.
- SubjectDN: The certificate gets created with `CN=<hostname>,OU=<client or server name>,O=ibm,C=US` as the SubjectDN.

The **createSSLCertificate** parameter can be called on the command line if users would like to customize the certificate.

**Note:** When the use of a collective controller is not practical, perhaps there is only one or two Liberty servers, a self-signed certificate can be used to restrict the number of clients that can connect to the Liberty member server. It is suggested that an IHS server is used in front of the Liberty servers, where an appropriate CA signed certificate can be used, along with CN whitelisting to control which clients can connect to HIS. A trusted channel between IHS and the Liberty Member server can be maintained by using the self signed certificate.

## Default SSL configuration

The minimal configuration that is needed for SSL is a single keystore element called `defaultKeyStore`. When the `defaultKeyStore` exists in the configuration the run time builds an SSL configuration that is called `defaultSSLConfig` around it.

`defaultSSLConfig` details:

- Protocol: When the IBM JRE is used, the protocol is set to `SSL_TLS` by default. If the Oracle JRE is used SSL is used as the protocol.
- Ciphers: The cipher list is built by getting a list of supported ciphers from the underlying JRE. By default the list is reduced to all the ciphers that are 128 bit and higher or 3DES. RC4 is removed because they are not considered safe to have enabled. ECDHE ciphers are removed because they can cause errors if you are going to a server that does not support them. The cipher list can be customized to include them.



- Client authentication: By default `clientAuthentication` and `clientAuthenticationSupported` are disabled.
- Keystore: In the default configuration `defaultKeyStore` is used as both the key and truststore.

An `ssl` element that is called `defaultSSLConfig` can be entered in the `server.xml` file for customization of the SSL configuration properties. A customized `ssl` element that is called `defaultSSLConfig` is still treated like the default SSL configuration as long as a different SSL configuration has not been identified as the default. For more information on attributes that can be used to customize an SSL configuration, see [SSL configuration attributes](#).

To designate a different `ssl` element in the configuration as the default SSL configuration, users can identify that with the `sslDefault` element.

```
<sslDefault sslRef="customSSLConfiguration" />
```

The attributes from the Liberty default SSL configuration are used to create an `SSLContext`. That `SSLContext` is set on the process as the default `SSLContext` by using the Java API `SSLContext.setDefault()`. If an application makes a call to an API like `URLConnection()` with an `https` URL and does not provide any SSL information, then the application picks up the default `SSLContext` of the process and in this case is the `SSLContext` that is created with the Liberty default SSL configuration.

If there is no default SSL configuration in Liberty, then the JSSE's default `SSLContext` is used. The JSSE's default `SSLContext` uses the `cacerts` file for the keystore and truststore. There is no default SSL configuration in Liberty if there is no SSL feature that is defined or if the SSL configuration that exists is not identified as the default. The default configuration is either the called `defaultSSLConfig`, which can be implicit if a `defaultKeyStore` is defined or an alternative SSL configuration can be designated by using the `sslDefault` element.

The `javax` system properties, `javax.net.ssl.keystore`, is used to set up the keystore and truststore information for the default SSL context and must not be used. If the properties are set on the process, then the call to `SSLContext.setDefault()` wipes them out.

## Creating SSL certificates for your Liberty using the Utilities menu

Using the Liberty Utilities menu in the developer tools, you can create an SSL certificate.

### Procedure

1. In the Servers view, right-click your Liberty server, and select **Utilities > Create SSL Certificate**.
2. On the Create SSL Certificate page, you can create a default secure socket layer (SSL) certificate to use with your server.
  - a. In the **Keystore password** field, type a password for your SSL certificate.
  - b. Click the **Specify validity period (days)** field, and specify the number of days you want the certificate to be valid for. Minimum length of time is 365 days.
  - c. Click the **Specify subject (DN):** field, and provide a value for your SSL subject.
3. Click **Finish**.

## Creating SSL certificates from the command line

You can use the `securityUtility` command to create a default SSL certificate for use by the Liberty configuration.

### Procedure

1. Open a command line, then change directory to the `wlp/bin` directory.
2. Create an SSL certificate.

Run the following command. If you do not specify a server name or a password, the command does not run. See “securityUtility command.”

```
securityUtility createSSLCertificate --server=server_name --password=your_password
```

## Results

You have created a default keystore `key.jks` for the specified server. The keystore file is located under the `/resources/security` directory of the specified server. If a default keystore already exists, the command does not execute successfully.

## What to do next

You can configure your server to use the keystore and enable the SSL in the server configuration by adding the following lines to the server configuration file:

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" password="keystore_password" />
```

See “Enabling SSL communication in Liberty” on page 1152.

## securityUtility command

The **securityUtility** command supports plain text encryption and SSL certificate creation for Liberty.

## Syntax

The command syntax is as follows:

```
securityUtility task [options]
```

Where the **options** are different based on the value of **task**.

## Parameters

The following tasks are available for the `securityUtility` command:

### encode

Encodes the provided *text* by using Base64. If no options are specified, the command enters interactive mode. Otherwise, the provided *text* is encoded. If the *text* includes spaces it must be put in quotation marks.

The options are:

#### **--encoding=encoding\_type**

Specifies how to encode the password. Supported encodings are `xor`, `aes`, and `hash`. If this option is not provided, the default is `xor`.

**Note:** The `hash` encoding option is used for encoding passwords for the basic user registry only.

#### **--key=encryption\_key**

Specifies the key to be used when encoding using AES encryption. This string is hashed to produce an encryption key that is used to encrypt and decrypt the password. The key can be provided to the server by defining the variable `wlp.password.encryption.key` whose value is the key. If this option is not provided, a default key is used.

#### **--notrim**

Specify whether space characters are removed from the beginning and end of the

specified text. If this option is specified, the provided text is encoded as it is. If this option is not specified, space characters from the beginning and end of the specified text is removed.

*text* The text that is to be encoded.

See also “The limits to protection through password encryption” on page 616.

### **createSSLCertificate**

Creates a default keystore including an SSL certificate for use in a server or client configuration.

8.5.5.6

#### **Keystore details:**

location: In the server's or client's directory under `resource/security/key.jks`.

type: JKS

password: Password provided with the `--password` option. The password is needed to open the keystore file and retrieve the key from the keystore file.

8.5.5.6

#### **Certificate details:**

type: Self-signed certificate

size: 2048 by default, alternate size can be specified with the `--keySize` option.

signature algorithm: SHA256withRSA, can be customized with the `--sigAlg` option.

validity: 365 days by default, can be customized with the `--validity` option.

SubjectDN: CN=<hostname>,OU=<client or server name>,O=ibm,C=us by default, can be customized with the `--subject` option.

The options are:

8.5.5.6

#### **--server=name**

Specifies the name of the Liberty server for which the keystore and certificate is created. This option cannot be used if the `--client` option is specified.

8.5.5.6

#### **--client=name**

Specifies the name of the Liberty client for which the keystore and certificate is created. This option cannot be used if the `--server` option is specified.

8.5.5.6

#### **--keySize=size**

Specifies the certificate key bit size. The default value is 2048.

#### **--password=password**

Specifies the password to be used in the keystore, which must be at least 6 characters in length. This option is required.

#### **--passwordEncoding=password\_encoding\_type**

Specifies how to encode the keystore password. Supported encoding value is `xor` or `aes`. If this option is not provided, a default value of `xor` is used.

#### **--passwordkey=password\_encryption\_key**

Specifies the key to use to encode the keystore password by using AES encryption. This string is hashed to produce an encryption key that is used to encrypt and decrypt the password. The key can be provided to the server by defining the variable `wlp.password.encryption.key` whose value is the key. If this option is not provided, a default key is used.

#### **--validity=days**

Specifies the number of days that the certificate is valid, which must be equal to or greater than 365. If this option is not provided, a default value of 365 is used.

#### 8.5.5.6 --subject=DN

Specifies the Distinguished Name (DN) for the certificate subject and issuer. If this option is not provided, a default value of CN=<hostname>,OU=<server or client name>,O=ibm,C=us is used. The CN value is retrieved by using a java method to get the machine's local host name. If the host name cannot be resolved, the IP address is returned.

#### 8.5.5.7 --sigAlg

Specifies the signature algorithm that is used to sign the self-signed certificate. The signature algorithm that is supported depends on what is supported by the underlying JRE. Stronger signature algorithms might require the JRE to have the unrestricted policy file in place.

The command accepts SHA256withRSA (default), SHA1withRSA, SHA384withRSA, SHA512withRSA, SHA1withECDSA, SHA256withECDSA, SHA384withECDSA, and SHA512withECDSA. The signature algorithms that end with RSA creates certificates with RSA keys and those that end with ECDSA creates certificates with Elliptical Curve (EC) keys.

**Note:** If you are using certificates that are created with EC keys, then your server needs a customized ciphers list in the ssl configuration to include EC ciphers.

**help** Prints help information for a specified task.

## Usage

The following examples demonstrate correct syntax:

```
securityUtility encode --encoding=aes GiveMeLiberty
securityUtility createSSLCertificate --server=myserver --password=mypassword --validity=365
 --subject=CN=mycompany,O=myOrg,C=myCountry
securityUtility help createSSLCertificate
```

### CAUTION:

**Different operating system might treat some characters differently. For the Windows environment, if you have ! in your input string, it needs to be escaped by the ^ character. For example,**

```
D:\Liberty\images\855\Liberty855\wlp\bin>securityUtility encode "a^!"
```

## Configuring your web application and server for client certificate authentication

You can configure your web application on Liberty using SSL client authentication.

### Before you begin

This topic assumes that you have already created the SSL certificates, for example as described in “Creating SSL certificates from the command line” on page 1161.

### About this task

Client certificate authentication occurs if the server-side requests that the client-side send a certificate. A WebSphere server can be configured for client certificate authentication on the SSL configuration. To do this, you add the ssl-1.0 Liberty feature to the server.xml file, along with code that tells the server the keystore information for authentication.

For details of which aspects of SSL are supported, see “Liberty features” on page 483.

## Procedure

1. Ensure that the deployment descriptor for your web application specifies client certificate authentication as the authentication method to use.

Check that the deployment descriptor includes the following element:

```
<auth-method>CLIENT-CERT</auth-method>
```

**Note:** You can use a tool such as Rational Application Developer to create the deployment descriptor.

2. Optional: Generate an SSL certificate using the command line. See “securityUtility command” on page 1162.
3. Configure your server to enable SSL client authentication by adding the following lines to the server.xml file:

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>

<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
<keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="defaultPWD" />
<keyStore id="defaultTrustStore" location="trust.jks" type="JKS" password="defaultPWD" />
```

- If you specify `clientAuthentication="true"`, the server requests that a client sends a certificate. However, if the client does not have a certificate, or the certificate is not trusted by the server, the handshake does not succeed.
  - If you specify `clientAuthenticationSupported="true"`, the server requests that a client sends a certificate. However, if the client does not have a certificate, or the certificate is not trusted by the server, the handshake might still succeed.
  - If you do not specify either `clientAuthentication` or `clientAuthenticationSupported`, or you specify `clientAuthentication="false"` or `clientAuthenticationSupported="false"`, the server does not request that a client send a certificate during the handshake.
4. Add a client certificate to your browser. See the documentation of your browser for adding client certificates.
  5. Make sure the server trusts any client certificates that are used.
  6. Make sure any client certificates used for client authentication are mapped to a user identity in your registry.
    - For the basic registry, the user identity is the common name (CN) from the distinguished name (DN) of the certificate.
    - For a Lightweight Directory Access Protocol (LDAP) registry, the DN from the client certificate must be in the LDAP registry.
  7. To use basic authentication, user ID and password only, if client certificate authentication does not succeed, add the following line to your server.xml file.

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

**Note:** If you specify `allowFailOverToBasicAuth="false"` or do not specify `allowFailOverToBasicAuth`, and the client certificate authentication does not succeed, the request generates a 403 Authentication error message, and the client is not prompted for basic authentication.

## Setting up Liberty to run in SP800-131a

You can set up Liberty to meet the SP800-131a requirement that is originated by the National Institute of Standards and Technology (NIST).

### About this task

SP800-131a requires longer key lengths and stronger cryptography. The specification also provides a configuration that enables users to move to a strict enforcement of SP800-131a. The configuration also

enables users to run with a mixture of settings from both FIPS140-2 and SP800-131a. SP800-131a can be run in two modes, transition and strict. The transition mode is offered to give user a setting to move their environment to SP800-131a strict mode. In transition mode, it is optional to use the SP800-131a required certificates and to set the protocol to SP800-131a

Strict enforcement of SP800-131a requirements on Liberty includes the following:

- The use of the TLSv1.2 protocol for the Secure Sockets Layer (SSL) context.
- Certificates must have a minimum length of 2048. Elliptical Curve (EC) certificate require a minimum size of 244-bit curves.
- Certificates must be signed with a signature algorithm of SHA256, SHA384, or SHA512. Valid signatureAlgorithms include:
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA

**Note:** If SHA384withECDSA or SHA512withECDSA is used, the unrestricted policy file needs to be in place for the IBM JDK.

- SP800-131a approved Cipher suites.

**Note:** To configure a Liberty server to run in SP800-131a mode, users must be running with a level of the IBM JDK that supports SP800-131a. The minimal levels of the IBM JDK include Java 6 sr 10, Java 6.0.1 sr 2, or Java 7.

For more information about the SP800-131a standard, see the National Institute of Standards and Technology.

You can configure Liberty to run in SP800-131a strict mode or transition mode as following:

## Procedure

- Configure Liberty to run in SP800-131a strict mode.
  1. Make sure that you are running on a level of the IBM JDK that supports SP800-131a.
  2. Make sure that certificates of your server meet the criteria for SP800-131a.
    - Certificates have a minimum length of 2048 and Elliptical Curve (EC) certificates have a minimum size of 244-bit curve.
    - Certificates are signed with at least SHA256 or signed with one of the signature algorithms listed previously.
  3. Configure your SSL Configuration to use the TLSv1.2 protocol. See “Enabling SSL communication in Liberty” on page 1152 and “SSL configuration attributes” on page 1153 for more details.
  4. When using collectives, if the sslProtocol is updated, two configuration changes must be made:
    - `${wlp.install.dir}/etc/server.env` must specify the `-Dhttps.protocols` property in order for the `${wlp.install.dir}/bin/collective` utility to successfully communicate with the controller.  
For example:  
`JVM_ARGS=-Dhttps.protocols=TLSv1.2`
    - Each internal collective replication ssl id must be updated with the desired protocol.  
For Example:  
`<ssl id="controllerConnectionConfig" sslProtocol="TLSv1.2"/>`  
`<ssl id="memberConnectionConfig" sslProtocol="TLSv1.2"/>`

- 5. Optional: If Elliptical Curve (EC) ciphers are required, list them in the **enabledCiphers** attribute. EC ciphers are not included when cipher lists are generated using the **securityLevel** attribute of the SSL Configuration. For the full list of ciphers, see the Java Technology Security information.
  - 6. The Java Secure Socket Extension (JSSE) is enabled to run in SP800-131a strict mode by setting the system property `com.ibm.jsse2.sp800-131` to *strict*. For example, `-Dcom.ibm.jsse2.sp800-131=strict`. See “Customizing the Liberty environment” on page 947 for how to set system properties in the `jvm.options` file.
- Configure Liberty to run in SP800-131a transition mode.
    1. Make sure that you are running a level of the IBM JDK that support SP800-131a.
    2. Optional: If Elliptical Curve (EC) ciphers are required, list them in the **enabledCiphers** attribute. EC ciphers are not included when cipher lists are generated using the **securityLevel** attribute of the SSL Configuration. For the full list of ciphers, see the Java Technology Security information.
    3. The JSSE is enabled to run in SP800-131a transition mode by setting the system property `com.ibm.jsse2.sp800-131` to *transition*. For example, `-Dcom.ibm.jsse2.sp800-131=transition`. See “Customizing the Liberty environment” on page 947 for how to set system properties in the `jvm.options` file.

**Note:** If you change your protocol to use TLSv1.2, make sure that your browser supports TLSv1.2.

## Configuring an `httpEndpoint` to use an SSL configuration other than the default

By default, an `httpEndpoint` element uses the server default SSL configuration, `defaultSSLConfig`. You can configure an `httpEndpoint` to use an SSL configuration other than the default SSL configuration.

### About this task

You can configure an `httpEndpoint` to use an SSL configuration in multiple ways. The following examples show different ways to configure an `httpEndpoint` to use an SSL configuration other than the default SSL configuration.

### Procedure

- Set the SSL options directly on the `httpEndpoint`. The following example shows how to set the SSL options on the `httpEndpoint`, and assumes that you have an SSL configuration that is named `wasListenerSSLConfig` already defined that is not in this example:
 

```
<httpEndpoint id="defaultHttpEndpoint"
 host="{listener.host}"
 httpPort="{http.port}"
 httpsPort="{https.port}">
 <sslOptions sslRef="wasListenerSSLConfig" />
</httpEndpoint>
```
- Reference an `sslOption` element in the `httpEndpoint`. The following example shows how to reference an `sslOption` element, and assumes that you have an SSL configuration that is named `wasListenerSSLConfig` already defined that is not in this example:
 

```
<sslOptions id="mySSLOptions" sslRef="wasListenerSSLConfig" />

<httpEndpoint id="defaultHttpEndpoint"
 host="{listener.host}"
 httpPort="{http.port}"
 httpsPort="{https.port}"
 sslOptionsRef="mySSLOptions"
/>
```
- Change the default `sslOptions` element to point to an SSL configuration other than the default SSL configuration. This option does not alter the `httpEndpoint`. The following example shows how to

change the default `sslOptions` element, and assumes that you have an SSL configuration that is named `wasListenerSSLConfig` already defined that is not in this example:

```
<sslOptions id="defaultSSLOptions" sslRef="wasListenerSSLConfig" />
```

---

## Authenticating users in Liberty

The Liberty server uses a user registry to authenticate a user and retrieve information about users and groups to perform security-related operations, including authentication and authorization.

### About this task

To learn about how authentication works in Liberty, see “Authentication” on page 585.

The authentication tasks that you can configure might vary depending on your requirements. Unless you have used the `quickStartSecurity` element that can configure only one user, you have to configure the user registry at the least. You do not have to configure the values for JAAS, authentication Cache and SSO tasks unless you want to change the default values. Configure TAI configuration only when you have an implementation of TAI interface to handle authentication.

You can complete one or more of the following authentication tasks:

### Procedure

- Configure authentication cache in Liberty
- Configure a custom JAAS login module for Liberty
- Configure SSO on Liberty
- Configure a user registry for Liberty
- Configure RunAS authentication in Liberty
- Configure TAI for Liberty

## Configuring a user registry for Liberty

You can store user and group information for authentication in several types of registries. For example you can use a basic user registry, an LDAP registry, or a Custom User Registry. Optionally, you can configure two or more LDAP registries so that the operations are executed on all the configured registries. For example, when you perform an operation of searching a user, the search is performed on all the configured LDAP registries.

### Procedure

- Configure a basic user registry for Liberty
- Configure an LDAP user registry for Liberty

## Configuring a basic user registry for Liberty

You can configure a basic user registry in Liberty for authentication.

### About this task

You can use a basic user registry by defining the users and groups information for authentication on the Liberty server. To do this, you add the `appSecurity-2.0` Liberty feature to the `server.xml` file, along with user information in the `basicRegistry` element.

### Procedure

1. Add the `appSecurity-2.0` Liberty feature to the `server.xml` file.
2. Optional: To use SSL, add the `ssl-1.0` Liberty feature in the `server.xml` file. See “Enabling SSL communication in Liberty” on page 1152.



### 3. Configure the basic registry for the server as follows:

```
<basicRegistry id="basic" realm="customRealm">
 <user name="mlee" password="p@ssw0rd" />
 <user name="rkumar" password="pa$$w0rd" />
 <user name="gjones" password="{xor}Lz4sLCgwLTs=" />
 <group name="students">
 <member name="mlee" />
 <member name="rkumar" />
 </group>
</basicRegistry>
```

#### Notes:

- You must use unique names for your users and groups.
- You should remove all trailing and leading spaces from the user and group names.
- If user ID or password contains characters other than US-ASCII, make sure that the file is saved by using UTF-8 character encoding.
- **Distributed operating systems** If you use the WebSphere Application Server Developer Tools for Eclipse, the password is encoded for you automatically.
- If you edit the `server.xml` file directly, you can use the **securityUtility encode** command to encode the password for each user. The `securityUtility` command-line tool is available in the `$INSTALL_ROOT/bin` directory. When you run the **securityUtility encode** command, you either supply the password to encode as an input from the command line or, if no arguments are specified, the tool prompts you for the password. The tool then outputs the encoded value. Copy the value output by the tool, and use that value for the password. For example, to encode the password `GiveMeLiberty`, run the following command:  
**securityUtility encode GiveMeLiberty**
- There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty.

## Configuring LDAP user registries in Liberty

You can configure one or more Lightweight Directory Access Protocol (LDAP) servers in Liberty for authentication.

### Before you begin

Ensure that your LDAP server is up and running, and that the host name and port number of the LDAP server are already in your known list.

### About this task

You can use an existing LDAP server for application authentication in Liberty. To do this, you add the `appSecurity-2.0` feature to the `server.xml` file and specify, in the `server.xml` file, the `ldapRegistry-3.0` feature, and the configuration information for connecting to the LDAP server.

**Avoid trouble:** There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty. For more information, see the link in the related reference for config snippets.

### Procedure

1. Add the `appSecurity-2.0` and `ldapRegistry-3.0` Liberty features to the `server.xml` file.
2. Optional: To communicate with an SSL-enabled LDAP server, add the `ssl-1.0` Liberty feature in the `server.xml` file.
3. Optional: Copy the truststore to the server configuration directory. For example, you can use the `${server.config.dir}` variable.

For SSL communication with an LDAP server to succeed, the Signer certificate for the LDAP server must be added to the truststore that is referenced by the **sslAlias** attribute of the <ldapRegistry> element. In the following examples, the Signer certificate must be added to the LdapSSLTrustStore.jks.

#### 4. Configure the LDAP entry for the server.

If you do not want SSL for the LDAP server, remove all SSL and keystore-related lines from the following examples.

You configure the LDAP server in the server.xml file or by using the WebSphere Application Server Developer Tools for Eclipse. There are several security configuration examples on the WASdev.net website for reference when configuring security for your applications on Liberty.

- For IBM Directory Server:

```
<ldapRegistry id="ldap" realm="SampleLdapIDSRealm"
 host="ldapserverserver.mycity.mycompany.com" port="389" ignoreCase="true"
 baseDN="o=mycompany,c=us"
 ldapType="IBM Tivoli Directory Server"
 sslEnabled="true"
 sslRef="LDAPSSLSettings">
 <idsFilters
 userFilter="(&(uid=%v)(objectclass=ePerson))"
 groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
 (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
 userIdMap="*:uid"
 groupIdMap="*:cn"
 groupMemberIdMap="mycompany-allGroups:member;mycompany-allGroups:uniqueMember;
 groupOfNames:member;groupOfUniqueNames:uniqueMember">
 </idsFilters>
</ldapRegistry>

<ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore" trustStoreRef="LDAPTrustStore" />

<keyStore id="LDAPKeyStore" location="{server.config.dir}/LdapSSLKeyStore.jks"
 type="JKS" password="{xor}CDo9Hgw=" />
<keyStore id="LDAPTrustStore" location="{server.config.dir}/LdapSSLTrustStore.jks"
 type="JKS" password="{xor}CDo9Hgw=" />
```

- For Microsoft Active Directory Server:

```
<ldapRegistry id="ldap" realm="SampleLdapADRealm"
 host="ldapserverserver.mycity.mycompany.com" port="389" ignoreCase="true"
 baseDN="cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
 bindDN="cn=testuser,cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
 bindPassword="testuserpwd"
 ldapType="Microsoft Active Directory"
 sslEnabled="true"
 sslRef="LDAPSSLSettings">
 <activatedFilters
 userFilter="(&(sAMAccountName=%v)(objectcategory=user))"
 groupFilter="(&(cn=%v)(objectcategory=group))"
 userIdMap="user:sAMAccountName"
 groupIdMap="*:cn"
 groupMemberIdMap="memberOf:member" >
 </activatedFilters>
</ldapRegistry>

<ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore" trustStoreRef="LDAPTrustStore" />

<keyStore id="LDAPKeyStore" location="{server.config.dir}/LdapSSLKeyStore.jks"
 type="JKS" password="{xor}CDo9Hgw=" />
<keyStore id="LDAPTrustStore" location="{server.config.dir}/LdapSSLTrustStore.jks"
 type="JKS" password="{xor}CDo9Hgw=" />
```

If you use the WebSphere Application Server Developer Tools for Eclipse, the bindPassword password is encoded for you automatically. If you edit the server.xml file directly, you can use the **securityUtility encode** command to encode the bindPassword password for you. The

securityUtility command-line tool is available in the \$INSTALL\_ROOT/bin directory. When you run the **securityUtility encode** command, you either supply the password to encode as an input from the command line or, if no arguments are specified, the tool prompts you for the password. The tool then outputs the encoded value. Copy the value output by the tool, and use that value for the bindPassword password.

- Optional: Configure certificate filter mode for the LDAP server.

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
 host="myldap.ibm.com" port="389" ignoreCase="true"
 baseDN="o=ibm,c=us"
 ldapType="IBM Tivoli Directory Server" searchTimeout="8m"
 certificateMapMode="CERTIFICATE_FILTER"
 certificateFilter="uid=${SubjectCN}">
 <idsFilters
 userFilter="(&(uid=%v)(objectclass=ePerson))"
 groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
 (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
 userIdMap="*:uid"
 groupIdMap="*:cn"
 groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;
 groupOfNames:member;groupOfUniqueNames:uniqueMember">
 </idsFilters>
</ldapRegistry>
```

For more information about certificate map mode in Liberty, see “LDAP certificate map mode” on page 1173.

- 8.5.5.3** Optional: You can define mapping between LDAP attributes and the user registry <externalId> attribute.

You can define mapping between LDAP attributes and the user registry <externalId> attribute. After the mapping is configured, when you use the user registry <externalId> attribute for any operation, the value will be equivalent to the value of the LDAP attribute that is mapped. The following example code shows the mapping that is defined for the user registry <externalId> attribute with the LDAP <distinguishedName> attribute for the entity type <PersonAccount>. The <autoGenerate> attribute is optional, and the value is false by default.

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
 host="myldap.ibm.com" port="389" ignoreCase="true"
 baseDN="o=ibm,c=us"
 ldapType="IBM Tivoli Directory Server" searchTimeout="8m">
 <attributeConfiguration>
 <externalIdAttribute name="distinguishedName" entityType="PersonAccount" autoGenerate="false"></externalIdAttribute>
 </attributeConfiguration>
</ldapRegistry>
```

- Optional: Configure failover for multiple LDAP servers.

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
 host="ldapserver1.mycity.mycompany.com" port="389" ignoreCase="true"
 baseDN="o=ibm,c=us" ldapType="IBM Tivoli Directory Server" idsFilters="ibm_dir_server">
 <failoverServers name="failoverLdapServersGroup1">
 <server host="ldapserver2.mycity.mycompany.com" port="389" />
 <server host="ldapserver3.mycity.mycompany.com" port="389" />
 </failoverServers>
 <failoverServers name="failoverLdapServersGroup2">
 <server host="ldapserver4.mycity.mycompany.com" port="389" />
 </failoverServers>
</ldapRegistry>

<idsLdapFilterProperties id="ibm_dir_server"
 userFilter="(&(uid=%v)(objectclass=ePerson))"
 groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
 (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
```

```

 userIdMap="*:uid" groupIdMap="*:cn"
 groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;
 groupOfNames:member;groupOfUniqueNames:uniqueMember">
</idsLdapFilterProperties>

```

For more information about the `ldapRegistry` and `failoverServers` elements, see \*\*\*\* MISSING FILE \*\*\*\*.

- Optional: Configure multiple LDAP registries. If multiple LDAP registries are configured in the `server.xml` file, they are federated automatically. Ensure that the users are unique across all federated repositories, otherwise the user registry operations are not successful.

**Note:** When you use multiple federated LDAP repositories, each repository must define a unique baseDN.

```

<ldapRegistry host="ldapsrv1.mycity1.mycompany.com" baseDN="o=mycompany,c=us"
 port="123" ldapType="IBM Tivoli Directory Server">
</ldapRegistry>

```

```

<ldapRegistry host="ldapsrv2.mycity2.mycompany.com"
 baseDN="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
 port="456"
 ldapType="Microsoft Active Directory"
 bindDN="cn=testuser,cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
 bindPassword="{xor}KzosKyos0i0vKDs=">
</ldapRegistry>

```

**Note:**

- Specifying the `federatedRepository` element is not mandatory to federate multiple LDAP registries because they are federated automatically. If the `federatedRepository` element is specified to configure the `participatingBaseEntry` and `primaryRealm` elements, then the user registry operations are performed only on the repositories that are defined in the `primaryRealm` element. You can define the input and output property mappings for different user registry APIs under the `primaryRealm` element.
- The `name` attribute of the `participatingBaseEntry` element must be the same as the value of `baseDN` attribute that is specified in the `ldapRegistry` element. In the example follows, the `baseDN` and `name` attributes are configured for the LDAP registry on the host `ldapsrv1.mycity1.mycompany.com`. The value of `baseDN` attribute must be the same as that of sub tree in your LDAP server and the value of `name` attribute must be the name of that sub tree in the federated user registry. It is optional to specify the `name` attribute. By default, the `name` attribute uses the same value as the `baseDN` attribute. If the `name` attribute is specified in the `ldapRegistry` element, then the `name` attribute in the `participatingBaseEntry` element must use the same value as the `name` attribute in the `ldapRegistry` element.

```

<ldapRegistry host="ldapsrv1.mycity1.mycompany.com" baseDN="o=mycompany,ou=myou,c=us"
 port="123" ldapType="IBM Tivoli Directory Server" name="o=mybaseentry">
</ldapRegistry>

```

```

<ldapRegistry host="ldapsrv2.mycity2.mycompany.com"
 baseDN="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
 port="456"
 ldapType="Microsoft Active Directory"
 bindDN="cn=testuser,cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
 bindPassword="{xor}KzosKyos0i0vKDs=">
</ldapRegistry>

```

```

<federatedRepository>
 <primaryRealm name="RealmName" delimiter="@" allowOpIfRepoDown="true">
 <participatingBaseEntry name="o=mybaseentry"/>
 <participatingBaseEntry name="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"/>
 <uniqueUserIdMapping inputProperty="uniqueName" outputProperty="uniqueName"/>
 <userSecurityNameMapping inputProperty="principalName" outputProperty="principalName"/>
 <userDisplayNameMapping inputProperty="principalName" outputProperty="principalName"/>
 </primaryRealm>
</federatedRepository>

```

```

 <uniqueGroupIdMapping inputProperty="uniqueName" outputProperty="uniqueName"/>
 <groupSecurityNameMapping inputProperty="cn" outputProperty="cn"/>
 <groupDisplayNameMapping inputProperty="cn" outputProperty="cn"/>
 </primaryRealm>
</federatedRepository>

```

For more information about the federated ldapRegistry elements, see \*\*\*\* MISSING FILE \*\*\*\*.

9. Optional: You can configure other optional attributes for the LDAP registry, such as **contextPool** or **ldapCache**, as given in the following example:

```

<ldapRegistry id="IBMDirectoryServerLDAP" realm="SampleLdapIDSRealm"
 host="host.domain.com" port="389" ignoreCase="true"
 baseDN="o=domain,c=us"
 bindDN="cn=testuser,o=domain,c=us"
 bindPassword="mypassword"
 ldapType="IBM Tivoli Directory Server"
 searchTimeout="8m">
 <contextPool enabled="true" initialSize="1" maxSize="0" timeout="0s" waitTime="3000ms" preferredSize="3"/>
 <ldapCache>
 <attributesCache size="4000" timeout="1200s" enabled="true" sizeLimit="2000"/>
 <searchResultsCache size="2000" timeout="600s" enabled="true" resultsSizeLimit="1000"/>
 </ldapCache>
</ldapRegistry>

```

**Note:**

- Federated user registry uses the context pooling mechanism to improve the performance of concurrent access to an LDAP server. Context pooling works at a higher level than the connection pooling. Each context entry in the context pool corresponds to a socket connection to the LDAP server. The bind credentials that are used by this pool are specified when you configure the LDAP registry.
- Federated repository uses the cache mechanism for performance enhancement. It caches information about the LDAP users and groups based on the user operations performed. For example, if you perform a search operation on the LDAP users and groups, the result of the operation is cached. You can enable the ldapCache element in the server.xml file as shown in the previous example.

**Troubleshooting tip:** **8.5.5.2** To troubleshoot any LDAP authentication issues, use the following trace specifications in the bootstrap.properties file:

```
com.ibm.ws.security.wim.*=all:com.ibm.websphere.security.wim.*=all
```

**LDAP certificate map mode:**

The certificate map mode is used to specify whether to map X.509 certificates into an LDAP directory by **EXACT\_DN** or **CERTIFICATE\_FILTER** in Liberty.

The **EXACT\_DN** means that the Distinguished Name (DN) in the certificate must exactly match the user entry in the LDAP server, including case and spaces. To use the specified certificate filter for the mapping, you can use the **CERTIFICATE\_FILTER**.

**Certificate filter**

Specifies the filter certificate mapping property for the LDAP filter. The filter is used to map attributes in the client certificate to entries in the LDAP registry.

If more than one LDAP entry matches the filter specification at run time, authentication fails because the result is an ambiguous match. The syntax this filter is:

```
LDAP attribute=${Client certificate attribute}
```

An example of a simple certificate filter is: uid=\${SubjectCN}.

You can also specify multiple properties and values as part of a certificate filter. The LDAP attribute of the filter specification depends on the schema that your LDAP server is configured to use. The client certificate attribute is one of the public attributes in your client certificate. The client certificate attribute must begin with a dollar sign, \$, and opening brace, {, and end with a closing brace, }. The attributes are case-sensitive.

The following LDAP attributes are supported:

- uid
- initials
- sAMAccountName
- displayName
- distinguishedName
- displayName
- description

The following client certificate attributes are supported:

- \${SubjectCN}
- \${SubjectDN}
- \${IssuerCN}
- \${IssuerDN}
- \${SerialNumber}

An example of an LDAP configuration with certificate filter mode enabled:

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
 host="myldap.ibm.com" port="389" ignoreCase="true"
 baseDN="o=ibm,c=us"
 certificateMapMode="CERTIFICATE_FILTER"
 certificateFilter="uid=${SubjectCN}"
 userFilter="(& (uid=%v) (objectclass=ePerson))"
 groupFilter="(& (cn=%v) (| (objectclass=groupOfNames)
 (objectclass=groupOfUniqueNames) (objectclass=groupOfURLs)))"
 userIdMap="*:uid"
 groupIdMap="*:cn"
 groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;
 groupOfNames:member;groupOfUniqueNames:uniqueMember"
 ldapType="IBM Tivoli Directory Server" searchTimeout="8m" />
```

## Configuring SCIM for user and group member management

8.5.5.8

You can configure the scim-1.0 feature in the `server.xml` file to enable user and group member management. System for Cross-domain Identity Management (SCIM) defines REST APIs to create, retrieve, update, and delete (CRUD) users and groups. Calls are made through a systems management REST WAB. The local calls will be HTTP over localhost through Web API only; no java APIs for local calls.

### Procedure

Adding the scim-1.0 feature in the `server.xml` file enables SCIM functions. But to complete the configuration, you must also perform the following configuration steps:

- **SSL Configuration:** The REST services are protected and can be accessed only on the HTTPS port. For more information about how to complete the SSL configuration, see “Enabling SSL communication in Liberty” on page 1152.

- Configuration of Federation Registry: The SCIM functions is only supported by the Federation Registry. To quickly set up a federation registry by using LDAP, see “Configuring LDAP user registries in Liberty” on page 1169.
- Configuration of an administrator role: The REST services are only accessible by an administrator, so a user needs to be configured with an administrator role. For more information about mapping the administrator role to Liberty, see “Mapping the administrator role for Liberty” on page 1023.

**Note:** For configuration of the administrator role for SCIM, you cannot use the Quick Start Registry.

- Configuration of HTTPS port (optional): The HTTP end point must be configured. For more information about the `httpEndpoint` feature element configuration, see the `httpEndpoint` section in “Admin Center” on page 508.

After the configuration steps are completed, the `scim-1.0` feature is now ready to be used. A sample configuration in the `server.xml` file is shown in the following example:

```
<server description="server1">
 <!-- Enable features -->
 <featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>servlet-3.0</feature>
 <feature>ldapRegistry-3.0</feature>
 <feature>scim-1.0</feature>
 <feature>ssl-1.0</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint" httpPort="9080" httpsPort="9090">
 <tcpOptions soReuseAddr="true" />
 </httpEndpoint>

 <ldapRegistry id="LDAP1" realm="SampleLdapIDSRealm" host="9.127.1.90" port="1389" ignoreCase="true"
 baseDN="o=ibm,c=us" ldapType="IBM Tivoli Directory Server" searchTimeout="8m" recursiveSearch="true"
 bindDN="cn=xxx" bindPassword="xxxxxx">
 <ldapEntityType name="PersonAccount">
 <rdnProperty name="uid" objectClass="inetOrgPerson"/>
 <objectClass>inetOrgPerson</objectClass>
 </ldapEntityType>
 <ldapEntityType name="Group">
 <objectClass>groupofnames</objectClass>
 <objectClass>ibm-nestedGroup</objectClass>
 <rdnProperty name="cn" objectClass="groupofnames"/>
 </ldapEntityType>
 <attributeConfiguration>
 <attribute name="title" propertyName="honorificPrefix" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="initials" propertyName="middleName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="st" propertyName="honorificSuffix" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="l" propertyName="homeStateOrProvinceName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="street" propertyName="homeStreet" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="postalAddress" propertyName="homeCity" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="postalCode" propertyName="homePostalCode" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="postOfficeBox" propertyName="homeCountryName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="departmentNumber" propertyName="photoURLThumbnail" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="description" propertyName="photoURL" syntax="String" entityType="PersonAccount">
 </attribute>
 </attributeConfiguration>
 </ldapRegistry>
</server>
```

```

 </attributeConfiguration>
 <groupProperties>
 <memberAttribute name="member" dummyMember="uid=dummy" objectClass="groupOfNames" scope="direct"/>
 <memberAttribute name="ibm-memberGroup" objectClass="ibm-nestedGroup" scope="direct"/>
 </groupProperties>
 </ldapRegistry>

 <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" />
 <keyStore id="defaultKeyStore" password="Liberty"/>

 <administrator-role>
 <user>wasadmin</user>
 </administrator-role>

 <federatedRepository>
 <primaryRealm name="WIMRegistry">
 <participatingBaseEntry name="o=ibm,c=us"/>
 </primaryRealm>
 </federatedRepository>
</server>

```

## SCIM operations in Liberty: 8.5.5.8

The System for Cross-domain Identity Management (SCIM) 1.1 specifications are supported in Liberty.

### Retrieving resources

WebSphere Application Server Liberty supports SCIM 1.1 specifications. For more information about the specification, see <http://www.simplecloud.info/>.

To retrieve a known resource, you must send a GET request to the configured HTTP endpoint. For example, `/Users/{id}` or `/Groups/{id}`.

The following example shows the operations against an LDAP registry.

```

 <ldapRegistry id="LDAP1" realm="SampleLdapIDSRealm" host="9.127.1.90" port="1389" ignoreCase="true"
 baseDN="o=ibm,c=us" ldapType="IBM Tivoli Directory Server" searchTimeout="8m" recursiveSearch="true"
 bindDN="xxxxxx" bindPassword="xxxxxx">
 <ldapEntityType name="PersonAccount">
 <rdnProperty name="uid" objectClass="inetOrgPerson"/>
 <objectClass>inetOrgPerson</objectClass>
 </ldapEntityType>
 <ldapEntityType name="Group">
 <objectClass>groupofnames</objectClass>
 <objectClass>ibm-nestedGroup</objectClass>
 <rdnProperty name="cn" objectClass="groupofnames"/>
 </ldapEntityType>
 <attributeConfiguration>
 <attribute name="title" propertyName="honorificPrefix" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="initials" propertyName="middleName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="st" propertyName="honorificSuffix" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="l" propertyName="homeStateOrProvinceName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="street" propertyName="homeStreet" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="postalAddress" propertyName="homeCity" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="postalCode" propertyName="homePostalCode" syntax="String" entityType="PersonAccount">
 </attribute>

```



```

 <attribute name="postOfficeBox" propertyName="homeCountryName" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="departmentNumber" propertyName="photoURLThumbnail" syntax="String" entityType="PersonAccount">
 </attribute>
 <attribute name="description" propertyName="photoURL" syntax="String" entityType="PersonAccount">
 </attribute>
 </attributeConfiguration>
 <groupProperties>
 <memberAttribute name="member" dummyMember="uid=dummy" objectClass="groupOfNames" scope="direct"/>
 <memberAttribute name="ibm-memberGroup" objectClass="ibm-nestedGroup" scope="direct"/>
 </groupProperties>
</ldapRegistry>

```

To retrieve a resource from the LDAP, you must send a GET request as `https://localhost:9090/ibm/api/scim/Users/uid=jsmith,o=ibm,c=us`.

## Querying resources

To query resources, you must send a GET request to the configured HTTP endpoint and specify the filter for searching. You can also specify the attributes parameter to return a subset of values from the returned resources and specify the paging and sorting parameters to organize the returned resources. The following examples show some of the filter options:

- Specify the parameter to retrieve subset of the values: For example: `https://localhost:9090/ibm/api/scim/Users?filter=givename sw "Jo"` retrieves all users whose name starts with "Jo".
- Specify the attributes parameter to retrieve subset of the values: For example: `https://localhost:9090/ibm/api/scim/Users?filter=givename sw "Jo" &attributes=username,emails&filter=name.familyname eq "Smith"` retrieves the email addresses of the user name that starts with "Jo" and whose family name is "Smith".
- Specify the paging and sorting parameters to retrieve organized resources. For example, `https://localhost:9090/ibm/api/scim/Users?filter=givename sw "Jo"&attributes=username&sortBy=username&startIndex=3&count=5` retrieves the third page of the sorted result with each page containing five user names that start with "Jo" and whose family name is "Smith".

### Notes:

- The underlying federated repositories determine whether the values that are specified in the filters are case-sensitive. In case of LDAP, the values are not case-sensitive by default, unless the attribute is mapped to a case-sensitive LDAP attribute.
- User name cannot be used in the search filter.
- When federating basic, SAF or custom registries that do not support the SCIM attributes, the filter pattern is always searched against the user name and only the user name attribute is returned.

## Creating resources

To create new resources, you must send a POST request to the resource endpoint, that is, Users or Groups. The POST content must contain the user json object and the HTTP header content-type must be set to `application/json`.

**Note:** The `application/xml` content type is not supported.

To create a user, the following request must be posted to `https://localhost:9090/ibm/api/scim/Users`

Content-Type: `application/json`

Content-Length: ...

```
{
```

```

"schemas":["urn:scim:schemas:core:1.0"],
"userName":"bjensen",
"externalId": "uid=bjensen,o=ibm,c=us",
"name":{
 "familyName":"Jensen",
 "givenName":"Barbara"
}
}

```

The request must contain all the attributes that are required to successfully create a User or Group in the user registry. For example, in a back-end Tivoli Directory Server LDAP, the minimum set of LDAP attributes needed to create a user would be uid, sn, and cn; so any request for creation of user must contain userName, givenName, and familyName. The create operation fails if there are any schema violation from the user registry.

The externalId attribute acts like an identifier and must be specified. The attribute format depends on the specification of back-end user registry. In the previous example, the back-end is an LDAP server with a baseEntry as o=ibm,c=us and the RDN property for the user is set as UID, so the externalId becomes uid=bjensen,o=ibm,c=us.

Based on the create request, the created user object is returned.

```

{
 "schemas":["urn:scim:schemas:core:1.0"],
 "id":"uid=bjensen,o=ibm,c=us",
 "userName":"bjensen",
 "externalId":"uid=bjensen,o=ibm,c=us",
 "name":{
 "formatted":"Barbara Jensen",
 "familyName":"Jensen",
 "givenName":"Barbara"
 },
 "meta":{
 "lastModified":"2015-09-15T14:30:11",
 "location":"https://localhost:9090/ibm/api/scim/Users/uid=bjensen",
 "created":"2015-09-15T14:30:11"
 }
}

```

When more than one LDAPs are configured and a create operation needs to be invoked, a default parent for the entity must be defined. The defined parent specifies the base entry under which the new entity would be created. The following example shows the configuration of a default parent for PersonAccount.

```

<federatedRepository>
 <primaryRealm name="WIMRegistry">
 <participatingBaseEntry name="o=ibm,c=us"/>
 <participatingBaseEntry name="o=ldap"/>
 </primaryRealm>
 <supportedEntityType>
 <defaultParent>o=ldap</defaultParent>
 <name>PersonAccount</name>
 </supportedEntityType>
</federatedRepository>

```

## Modifying resources

To modify resources, one needs to send a PUT request to the resource endpoint, that is, /Users or /Groups. A PUT request performs a complete update of the resource. Any attributes that is not specified in the input are deleted.

**Note:** Partial updates with PATCH are not supported.

To modify the previously created resource, the following request must be posted to `https://localhost:9090/ibm/api/scim/Users/uid=bjensen,o=ibm,c=us`.

Content-Type: application/json  
Content-Length: ...

```
{
 "schemas":["urn:scim:schemas:core:1.0"],
 "userName":"bjensen",
 "externalId":"uid=bjensen,o=ibm,c=us",
 "name":{"
 "familyName":"Jensen",
 "givenName":"Barb"
 }}
}
```

This request modifies the given name of the object from *Barbara* to *Barb*. The ID specified in the URL must match the `externalId` specified in the user object.

To modify a Group `cn=employeeGroup,o=ibm,c=us` and change the Group membership, the `member` attribute needs to be specified.

Content-Type: application/json  
Content-Length: ...

```
{
 "id":"cn=employeeGroup,o=ibm,c=us",
 "schemas":["urn:scim:schemas:core:1.0"],
 "displayName":"employeeGroup",
 "externalId":"cn=employeeGroup,o=ibm,c=us",
 "members":[{"value":"uid=bjensen,o=ibm,c=us", "type":"User"}, {"value":"cn=consultants,o=ibm,c=us", "type":"Gr"}
]
```

To modify a Group `cn=employeeGroup,o=ibm,c=us` and remove all the members from a group an empty `member` attribute needs to be specified.

Content-Type: application/json  
Content-Length: ...

```
{
 "id":"cn=employeeGroup,o=ibm,c=us",
 "schemas":["urn:scim:schemas:core:1.0"],
 "displayName":"employeeGroup",
 "externalId":"cn=employeeGroup,o=ibm,c=us",
 "members":[]
}
```

## Deleting resources

To delete resources, you need to send a DELETE request to the resource endpoint, that is, `/Users` or `/Groups`. For example, to delete the user `uid=bjensen,o=ibm,c=us` that is created in the previous example, send a request to `https://localhost:9090/ibm/api/scim/Users/uid=bjensen,o=ibm,c=us`

**Note:** SCIM runs only on Java 7 or later.

## Configuring additional properties for users and groups

8.5.5.8

You can configure additional properties for users and groups of federated repositories. To enable schema or property extensions, ensure that the property can be read from and written to the underlying repositories.

## Procedure

1. You can specify the following additional property information in the `server.xml` file to enable schema or property extension.
  - Extended Property Name – The name of the extended property. Ensure that the name specified is unique and does not match with an existing property name.
  - Data type – The data type of the extended property. The possible values are Integer, Long, String, Boolean, Date, Double, BigInteger, BigDecimal.
  - Entity type – The entity to which the property applies. The possible values are PersonAccount or Group.

Single or multi-valued - You can set the value of the property to be either single or multi-valued. A default value can also be set for the property. When an entity is created and no value is specified for the property, the default value is used. For a multi-valued property, you can add an extended property named `assetId` for storing assets assigned to a user. If each user can be assigned more than one assets then the `assetId` needs to be multi-valued. You must ensure that the attribute to which the `assetId` is mapped is also a multi-valued attribute in the back-end LDAP .

The following sample shows the configuration in `server.xml`:

```
<federatedRepository>
 <primaryRealm name="WIMRegistry">
 <participatingBaseEntry name="o=ibm,c=us"/>
 </primaryRealm>
 <extendedProperty dataType="String" name="extendedProperty" entityType="PersonAccount"> </extendedProperty>
</federatedRepository>
```

2. To use the extended property in the code, you must use the generic getter/setter methods as shown in the following example:

```
PersonAccount person = new PersonAccount();
...
person.set("extendedProperty", "xyz");
...
String value = (String)person.get("extendedProperty");
```

3. To ensure that property can be read from and written to the LDAP, you have the following two options:
  - Pass-through: If the name of the extended property is same as the name of the LDAP attribute, then the property is passed through and read from and written to the attribute.
  - Property Mapping: If the name of the extended property is different from the name of the LDAP attribute, then the property needs to be mapped by using attribute mapping.

The following sample configuration shows the mapping of the extended property to an attribute named *extendedAttribute*.

```
<attributeConfiguration>
 <attribute name="extendedAttribute" propertyName="extendedProperty" syntax="String" entityType="PersonAccount"></attributeConfiguration>
```

## Dynamic changes to security

Some specific information on how dynamic configuration changes impact security on Liberty is introduced in this topic.

## Dynamically changing the user registry

Changing the user registry can impact both the server configuration and clients using the server. Before you change the user registry dynamically (without restarting the server), consider the following:

- If you change the user registry type or realm name, all web clients must clear their single sign-on tokens.
- If you change the user registry type or realm name, any values of **accessId** that are specified in the authorization bindings must be updated. The **accessId** takes the form of `user:realmName/uniqueId` or `group:realmName/uniqueId`. The **realmName** in the **accessId** must match the **realmName** for the configuration user registry.

## Developing a custom user repository for Liberty

8.5.5.8

You can develop a custom user repository class by implementing the `com.ibm.ws.security.wim.RepositoryFactory` and `com.ibm.ws.security.wim.Repository` interfaces that are provided in the Liberty server.

### About this task

The repository interfaces enable support to virtually any type of account repository.

### Procedure

1. Implement the repository factory (`com.ibm.ws.security.wim.RepositoryFactory`) interface. The `RepositoryFactory` interface gets the configuration parameters and creates an instance of the repository. For example,

```
public Repository getRepository(Map<String, Object> properties) throws WIMException {
 return new CustomRepository(properties);
}
```

2. Implement the repository (`com.ibm.ws.security.wim.Repository`) interface. This class has the actual repository operations.

```
public class CustomRepository extends RepositoryConfiguration implements Repository {

 public CustomRepository(Map<String, Object> properties) {
 System.out.println("Constructor with " + properties);
 }
}
```

3. Convert the implementation class into an OSGi service. For more information, see “Declaring your services to OSGi Declarative Services” on page 1116.
4. Package the custom user repository as an OSGi bundle and export the user repository service. For more information about creating an OSGi bundle, see [Creating an OSGi service bundle](#).
5. Create a feature manifest to include the OSGi bundle. For more information, see “Product extension” on page 577.
6. After the feature is installed into the user product extension location, configure the `server.xml` file with the feature name. For example:

```
<featureManager>
...
<feature>usr:customRepositorySample-1.0</feature>
</featureManager>
```

For a downloadable custom user registry sample, see [https://developer.ibm.com/wasdev/downloads/#asset/samples-Custom\\_User\\_Registry](https://developer.ibm.com/wasdev/downloads/#asset/samples-Custom_User_Registry).

### Example

Refer to the following samples of repository factory and repository interfaces.

#### Repository factory interface

```
package com.myorg;

import java.util.Map;

import org.osgi.service.component.ComponentContext;

import com.ibm.websphere.security.wim.exception.WIMException;
import com.ibm.ws.security.wim.Repository;
import com.ibm.ws.security.wim.RepositoryFactory;

public class CustomRepositoryFactory implements RepositoryFactory {
```

```

@Override
public Repository getRepository(Map<String, Object> properties) throws WIMException {
 System.out.println("getRepository " + properties);
 return new CustomRepository(properties);
}

public void activate(ComponentContext cc, Map<String, Object> properties) {
 System.out.println("In activate");
}

public void deactivate(ComponentContext cc) {
 System.out.println("In deactivate");
}
}

```

### Repository interface

```

package com.myorg;

import java.util.Map;

import com.ibm.websphere.security.wim.exception.WIMException;
import com.ibm.websphere.security.wim.model.Root;
import com.ibm.ws.security.wim.Repository;
import com.ibm.ws.security.wim.RepositoryConfiguration;

public class CustomRepository extends RepositoryConfiguration implements Repository {

 public CustomRepository(Map<String, Object> properties) {
 System.out.println("Constructor with " + properties);
 }

 @Override
 public Root create(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
 }

 @Override
 public Root delete(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
 }

 @Override
 public Root get(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
 }

 @Override
 public String getRealm() {
 return "customRepository";
 }

 @Override
 public Root login(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
 }

 @Override
 public Root search(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
 }
}

```

```

@Override
public Root update(Root arg0) throws WIMException {
 throw new WIMException("Method not supported");
}
}

```

## Configuring the authentication cache in Liberty

You can modify how authenticated users are cached in Liberty.

### About this task

Because the creation of a subject might impact performance, Liberty provides an authentication cache to store a subject after an authentication of a user is successful. The cache is initialized with a certain number of entries, determined by the **initialSize** attribute, and has a maximum number of entries, determined by the **maxSize** attribute. If the maximum size is reached, then the earliest entries that were used are removed from the cache. If a user has been inactive for longer than period that is specified by the **timeout** attribute, then the entry for that user is removed from the cache. By default, the cache size is initialized to 50 entries and a maximum of 25000 entries, with a timeout of 600 seconds.

You do not have to configure the values for the `authCache` element unless you want to change the default values of the authentication cache.

For more information about authentication case, see “Authentication cache” on page 587.

#### Note:

- Any change that is made to the user registry configuration in the `server.xml` file clears the authentication cache. However, if changes are made to an external user registry, such as LDAP, the authentication cache is unaffected.
- You must consider the following effects of the timeout value on your configuration:
  - Larger authentication cache timeout values can increase security risks. For example, you might revoke a user in the user registry or repository, but the revoked user can log in by using the credential that is cached in the authentication cache until the cache is refreshed.
  - Smaller authentication cache timeout values can affect performance. When this value is smaller, the Liberty server accesses the user registry or repository more frequently.
  - Larger numbers of entries in the authentication cache, which is caused by an increased number of users, increases the memory usage of the authentication cache. Thus, the application server might slow down and affect performance.

### Procedure

1. Enable the `appSecurity-2.0` Liberty feature by adding the following code to the `server.xml` file.

```

<featureManager>
 <feature>appSecurity-2.0</feature>
</featureManager>

```

2. Optional: To change the default options for the authentication cache, add the **<authCache>** element to the `server.xml` file. In the following example, the initial size of the authentication cache is changed to 100 entries with a maximum of 50000 entries, and the timeout is changed to 15 minutes.

```

<authCache initialSize="100" maxSize="50000" timeout="15m"/>

```

3. Optional: To disable the authentication cache, set the attribute **cacheEnabled** to `false` in the `<authentication>` element as follows:

```

<authentication id="Basic" cacheEnabled="false" />

```

For more information about the **<authCache>** and **<authentication>** elements, see \*\*\*\* MISSING FILE \*\*\*\*.

## Configuring a JAAS custom login module for Liberty

You can configure a custom Java Authentication and Authorization Service (JAAS) login module before or after you have configured the Liberty server login module.

### Before you begin

**8.5.5.9** We support the `server.xml` file, `client.xml` file and the JAAS configuration file for JAAS configuration. However, it is suggested to configure the JAAS custom login module in the `server.xml` file or `client.xml` file. For further details about configuring the JAAS configuration file, see “Configuring an application JAAS custom login context entry and login module using a JAAS configuration file for Liberty” on page 1185.

Make sure you have a JAR file containing the JAAS custom login module, which implements the `javax.security.auth.spi.LoginModule` interface as described in “Developing JAAS custom login modules for a system login configuration” on page 1305. In this topic, JAAS custom login module uses hashtable, callbacks or shared state variables provided by the Liberty server to pass authentication data to the system login module.

### About this task

You can use a custom login module to either make additional authentication decisions or add information to the subject to make finer-grained authorization decisions inside your application. See “JAAS configuration” on page 587 and “JAAS login modules” on page 588 for a more detailed overview.

You can also use the developer tools to configure a custom JAAS login module. **Distributed operating systems** See “Configuring JAAS on Liberty by using developer tools” on page 1186. There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty. See “Configuring JAAS on Liberty by using developer tools” on page 1186.

To configure a JAAS custom login module, complete the following steps:

### Procedure

1. Enable the `appSecurity-2.0` Liberty feature in the `server.xml` file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
</featureManager>
```
2. Create a class `com.sample.CustomLoginModule` that implements the `LoginModule` interface and package it into the `CustomLoginModule.jar` file.
3. Create a `<library>` element that uses a `<fileset>` element indicating where the `CustomLoginModule.jar` file is. In this example, the library `id` is `customLoginLib`.

```
<library id="customLoginLib">
 <fileset dir="{server.config.dir}" includes="CustomLoginModule.jar"/>
</library>
```
4. Create a `<jaasLoginModule>` element. In this example, the `id` is `custom`.
  - a. Configure the custom login module to require a successful authentication by setting the `controlFlag` attribute to `REQUIRED`.
  - b. Set the `libraryRef` attribute to `customLoginLib`, the `id` of the `<library>` element configured in the previous step. This login module also has two options: `UserRegistry` is `ldap` and `mapToUser` is `user1`.

```
<jaasLoginModule id="myCustom"
 className="com.sample.CustomLoginModule"
 controlFlag="REQUIRED" libraryRef="customLoginLib">
 <options myOption1="value1" myOption2="value2"/>
</jaasLoginModule>
```



5. Create a `<jaasLoginContextEntry>` element with an **id** and a unique **name** of the system-defined JAAS configuration: `system.WEB_INBOUND`. You can also set this JAAS configuration to `system.DEFAULT`, `WSLogin`, or your own JAAS configuration. On the `loginModuleRef` attribute, add `custom`, the **id** of the `jaasLoginModule` element created in the previous step. Putting this **id** first in the list means that it is the first JAAS login module to be called. You must also list the other default login modules: **hashtable**, **userNameAndPassword**, **certificate**, and **token**.

```
<jaasLoginContextEntry id="system.WEB_INBOUND" name="system.WEB_INBOUND"
 loginModuleRef="myCustom, hashtable, userNameAndPassword, certificate, token" />
```

**Note:** The option name cannot start with a period (`.`), **config.**, or **service** and must be unique. Also, the property name **id** or **ID** is not allowed.

For more information about the `<jaasLoginContextEntry>`, `<jaasLoginModule>`, `<options>`, and `<library>` elements, see **\*\*\*\* MISSING FILE \*\*\*\***.

## Configuring an application JAAS custom login context entry and login module using a JAAS configuration file for Liberty

8.5.5.9

JAAS configuration information can be configured in a JAAS configuration file.

### About this task

We support the `server.xml` file, `client.xml` file and the JAAS configuration file for JAAS configuration. However, it is suggested to configure the JAAS custom login module in the `server.xml` file or `client.xml` file. For further details about configuring the JAAS custom login module, see “Configuring a JAAS custom login module for Liberty” on page 1184.

The Liberty server reads the JAAS configuration file for an application JAAS custom login context entry and login module. The changes that are made to the JAAS configuration file are used by the local application and take effect after the application server is restarted. The JAAS configuration in the `server.xml` file takes precedence over what is defined in the JAAS configuration file. A configuration entry in the JAAS configuration file is overridden by an entry of the same alias name in the `server.xml` file.

To configure a JAAS custom login module, complete the following steps:

### Procedure

1. Enable the `appSecurity-2.0` Liberty feature in the `server.xml` file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 ...
</featureManager>
```

2. Create a JAAS custom login module class.

For example, `com.sample.CustomLoginModule` that implements the `LoginModule` interface and package it into the `CustomLoginModule.jar` file.

3. Create the default `jaas` directory.

#### For the server

```
${server.config.dir}/resources/security/jaas
```

#### For the client

```
${client.config.dir}/resources/security/jaas
```

**Note:** All JAAS custom login modules that are specified in the JAAS configuration file must place in the default `jaas` directory.

4. Place the `CustomLoginModule.jar` file in the default `jaas` directory.

5. Create a JAAS configuration file.

For example, create a `myJaas.conf` file and place it in the `${server.config.dir}/resources/security/jaas` directory that has the following content:

```
myCustomLoginContext {
 com.sample.CustomLoginModule required myOption1="value1" myOption2="value2"
};
```

6. Configure the JAAS configuration file using the `jvm.options` file. For example,  
`-Djava.security.auth.login.config=${server.config.dir}/resources/security/jaas/myJaas.conf`

**Note:** We only support the application custom JAAS login module in the JAAS configuration file. Do not put the default system JAAS configuration information in the JAAS configuration file.

**Note:** The JAAS configuration file is not dynamically updated if you made any changes. We strongly recommend configuring the JAAS configuration information in the `server.xml` file or `client.xml` file.

## Configuring JAAS on Liberty by using developer tools

### Distributed operating systems

You can configure a JAAS configuration (`system.WEB_INBOUND`) with a custom login module for Liberty by editing the configuration. You do not have to configure JAAS unless you want to customize it.

### Before you begin

For a description of the underlying process of configuring a server, and detailed information about specific aspects of server configuration, see “Administering Liberty manually” on page 946.

**Avoid trouble:** The developer tools creates the reference to a JAAS login module using the `loginModuleRef` element. You must change it and use the `loginModuleRef` attribute of `jaasLoginContextEntry` element. There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty.

### Procedure

1. Select **JAAS Login Context Entry** and click **Add**, then enter the login module names.
2. Select **JAAS Login Module: myCustom** and configure your custom login module by entering the **ID** and the **Class name**, then click the arrow next to the **Add** button and select **Global Element** to enter the shared library information.
3. Enter the **ID** for the shared library in the pop-up panel and click **OK**.
4. Configure **Name** and **Description** fields for the shared library, then click the arrow next to the **Add** button and select **Child Element** to add a **Fileset** reference as a child element.
5. Configure the **Fileset**. Click **Browse** in the **Base Directory** field and select the directory where the JAR file is located. Then, click **Browse** in the **Includes pattern** field to select your JAR file that contains your custom login module implementation.
6. Optional: If your custom login module needs any options, you can right-click **JAAS Login Module**, select **Add** and then select **login module options**.
7. Save the configuration. You can find the following configuration saved in the `server.xml` file.

```
<jaasLoginContextEntry name="system.WEB_INBOUND" id="system.WEB_INBOUND">
 <loginModuleRef>myCustom, hashtable, userNameAndPassword, certificate, token</loginModuleRef>
</jaasLoginContextEntry>

<jaasLoginModule className="com.sample.CustomLoginModule"
 id="myCustom" libraryRef="customLoginLib">
</jaasLoginModule>
```

```
<library id="customLoginLib" name="customLoginLib"
 description="Custom login module shared library">
 <fileset dir="{server.config.dir}" includes="CustomLoginModule.jar"/>
</library>
```

- Required:** To make the configuration work, you must change the `jaasLoginContextEntry` element to include the `loginModuleRef` attribute. You must remove the `loginModuleRef` element and add it as an attribute of the `jaasLoginContextEntry` element.

Here is an example of configuration using the `loginModuleRef` attribute.

```
<jaasLoginContextEntry name="system.WEB_INBOUND" id="system.WEB_INBOUND"
 loginModuleRef="myCustom, hashtable, userNameAndPassword, certificate, token" />

<jaasLoginModule className="com.sample.CustomLoginModule"
 id="myCustom" libraryRef="customLoginLib">
</jaasLoginModule>

<library id="customLoginLib" name="customLoginLib"
 description="Custom login module shared library">
 <fileset dir="{server.config.dir}" includes="CustomLoginModule.jar"/>
</library>
```

## Configuring a Java Authentication SPI for Containers (JASPIC) User Feature



8.5.5.6

You can develop a JASPIC provider to authenticate inbound web requests by using the `com.ibm.wsspi.security.jaspi.ProviderService` interface that is provided in the Liberty server.

### About this task

The Java Authentication SPI for Containers specification, JSR 196, defines an interface for authentication providers. In the Liberty server, you must package your JASPIC provider as a user feature. Your feature must implement the `com.ibm.wsspi.security.jaspi.ProviderService` interface.

### Procedure

- Create an OSGi component that provides a service that implements the `com.ibm.wsspi.security.jaspi.ProviderService` interface.

The `ProviderService` interface defines method, `getAuthConfigProvider`, which the Liberty runtime invokes to retrieve an instance of your JASPIC provider class that implements the `javax.security.auth.message.config.AuthConfigProvider` interface.

The following example uses OSGi declarative services annotations:

```
@package com.mycompany.jaspi;

import java.util.Map;
import javax.security.auth.message.config.AuthConfigFactory;
import javax.security.auth.message.config.AuthConfigProvider;
import org.osgi.service.component.ComponentContext;
import com.mycompany.jaspi.SampleAuthConfigProvider;
import com.ibm.wsspi.security.jaspi.ProviderService;

@Component(service = { ProviderService.class },
 configurationPolicy = ConfigurationPolicy.IGNORE,
 immediate = true,
 property = { "myProviderProperty1=value1",
 "myProviderProperty2=value2"})
public class SampleJaspiProviderService implements ProviderService {

 Map<String, String> configProps = null;
```

```

// This method called by the Liberty runtime
// to get an instance of AuthConfigProvider
@Override
public AuthConfigProvider getAuthConfigProvider(Map<String, String>
 AuthConfigFactory factory)
{
 return new SampleAuthConfigProvider(configProps, factory);
}

protected void activate(ComponentContext cc) {
 // Read provider config properties here if needed,
 // then pass them to the AuthConfigProvider factory.
 // This example reads the properties from the OSGi
 // component definition.
 configProps = (Map<String, String>) cc.getProperties();
}

protected void deactivate(ComponentContext cc) {}
}

```

2. Package the component into an OSGi bundle that is part of your user feature, along with your JASPIC authentication provider.
3. Ensure that your feature includes the OSGi subsystem content: `com.ibm.websphere.appserver.jaspic-1.1; type="osgi.subsystem.feature"`.
4. After the feature is installed into the user product extension location, configure the `server.xml` file with the feature name. For example:

```

<featureManager>
 ...
 <feature>usr:myJaspiProvider</feature>
</featureManager>

```

## Configuring LTPA in Liberty

You can configure a Liberty server to use a specific Lightweight Third Party Authentication (LTPA) keys file, user-defined password, and expiration time.

### About this task

The LTPA is configured by default when security is enabled for a Liberty server for the first time. The default location of the automatically generated LTPA keys file is `${server.output.dir}/resources/security/ltpa.keys`. The LTPA keys are encrypted with a randomly generated key and a default password of WebAS is initially used to protect the keys. The password is required when importing the LTPA keys into another server. To protect the security of the LTPA keys, you must change the password. When the LTPA keys are exchanged between servers, this password must match across the servers for Single Sign On (SSO) to work.

The default expiration timeout is 120 minutes. The expiration value refers to how long the LTPA tokens are valid before they expire.

To enable dynamic reloading of the LTPA keys when copying an LTPA keys file from another server, you can specify a file monitor interval before copying the LTPA keys file. The monitor interval value refers to how often the LTPA keys file is monitored for updates.

For more information about LTPA, see [LTPA concept in Liberty](#).

### Procedure

1. Configure the `<ltpa>` element in the `server.xml` file as follows, replacing the sample values in the example with your values:

```

<ltpa keysFileName="yourLTPAKeysFileName.keys" keysPassword="keysPassword" expiration="120" />

```

- Optional: Set the **monitorInterval** attribute to check the `ltpa.keys` file for key changes to be dynamically reloaded. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). In the following example, the LTPA keys file is checked for changes to be dynamically reloaded every 5 seconds:

```
<ltpa keysFileName="yourLTPAKeysFileName.keys" keysPassword="keysPassword"
 expiration="120" monitorInterval="5s" />
```

- Encode the password within the configuration. You can get the encoded value by using the `securityUtility encode` command.
- Optional: Copy an existing LTPA keys file to the location specified in the **keysFileName** attribute. The default value is `${server.output.dir}/resources/security/ltpa.keys`.  
For more information on `<ltpa>` element, see **\*\*\*\* MISSING FILE \*\*\*\***.

## OpenID



8.5.5.4

OpenID is an open standard where users can authenticate themselves to multiple entities without the need to manage multiple accounts or sets of credentials. WebSphere Application Server Liberty supports OpenID 2.0 and plays a role as a Relying Party in web single-sign-on.

### OpenID Provider (OP)

An OpenID Authentication server that can assert whether a user controls a unique identifier.

### Relying Party (RP)

An entity that requires proof that a user controls a unique identifier.

### OpenID Identifier:

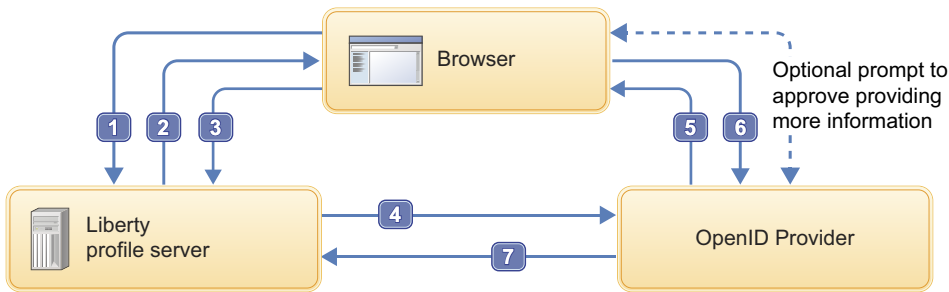
An http or https URL that belongs to an OpenID provider or a user.

Accessing various entities such as websites often requires a unique account that is associated with each entity. OpenID enables a single set of credentials that are handled by an OpenID Provider to grant access to any number of entities that support OpenID.

When required to sign in to an entity, such as a website that supports OpenID and acts as a Relying Party, users perform authentication by interacting directly with an OP rather than providing their credentials to the RP itself. An OP verifies the identity of the user and sends an authentication confirmation back to the RP. When this confirmation is received from the OP, the RP accepts the user as authenticated.

A typical OpenID authentication flow is described as follows:

- A user attempts to access a protected resource, such as a web page.
- The Liberty server, acting as a RP, presents a form login page for the protected resource.
- The user enters an OpenID identifier.
- The RP takes the identifier and redirects the user to the appropriate OP.
- The OP prompts the user for credentials.
- The user enters credentials for the account that is associated with the OP.
- The OP authenticates the user and optionally prompts the user to approve or deny providing user information to the RP, and subsequently redirects the user back to the RP with the authentication result.
- If the OP authentication is successful, the RP attempts to authorize the user.
- If the user authorization is successful, the RP establishes an authenticated session with the user.



OpenID helps minimize the chances of user credentials or sensitive information being mishandled. With OpenID, a user's credentials are only exchanged with the OpenID Provider, meaning no website other than the OP ever sees the user's credentials. This standard helps mitigate the possibility of an unscrupulous or insecure website compromising the identity of a user. The user also controls how much personal information is shared with the websites visited. For example, users can choose to allow the name and email address that are associated with their OpenID account to be shared with one website while electing not to provide their email address, or any information at all, to another website.

The OpenID standard specifications that are supported include:

OpenID Authentication 2.0.

OpenID Attribute Exchange 1.0.

**Note:** The OpenID 2.0 Claimed Identifier should be used to identify a user per spec, Identifying the end user. However, the Claimed Identifier is not user friendly and many Relying Parties choose one of the OpenID attributes to represent the user. The most popular attribute used to represent the user is a user's email address.

It is known that some OpenID attributes, including email, might not be validated by the OpenID provider. If you do not trust that a provider will provide you a verified email address, then you must not use the provider's email as the authenticated user name in the WebSphere Application Server.

## OpenID Connect



8.5.5.4

OpenID Connect is a simple identity protocol and open standard that is built on top of the OAuth 2.0 protocol that enables client applications to rely on authentication that is performed by an OpenID Connect Provider to verify the identity of a user.


OpenID Connect uses OAuth 2.0 for authentication and authorization and then builds identities that uniquely identify users. Client applications can also obtain basic profile information about a user in an interoperable and REST-like manner from OpenID Connect Providers.

The WebSphere Application Server Liberty supports OpenID Connect 1.0 and plays a role as a Client, or Relying Party, and as a Provider in web single sign-on. The following OpenID Connect specification is supported:



8.5.5.4

OpenID Connect Core 1.0

For those using a Liberty server as a Web-based Relying Party, the  **8.5.5.4** OpenID Connect Basic Client Implementer's Guide 1.0 is a subset of the OpenID Connect Core specification that is easier to read and provides details for Web-based Relying Parties that use the Authorization Code Flow.

**Access Token**

A credential that is used to access protected resources. An access token is a string that represents an authorization that is issued to the client.

**Authorization Endpoint**

A resource on an OpenID Provider that accepts an authorization request from a client to perform authentication and authorization of a user. The authorization endpoint returns an authorization grant, or code, to the client in the Authorization Code Flow. In the Implicit Flow, the authorization endpoint returns an ID token and access token to the client.

**Authorization Grant**

A credential that represents a user's authorization to access resources, and is used by a client to obtain an access token.

**Claim** Information asserted about an entity. Examples of a claim include phone number, first name, last name, and others.

**ID Token**

JSON Web Token (JWT) that contains claims about the authenticated user.

**Introspection Endpoint**

A resource on an OpenID Provider that enables a client that is holding an access token to retrieve information that was used to create the access token, such as the user name, granted scopes, client ID, or other information.

**OpenID Connect Provider (OP)**

An OAuth 2.0 authorization server that is capable of providing claims to a client, or Relying Party (RP).

**Refresh Token**

Issued to the client by the OP and is used to obtain a new access token when the current access token expires, or to obtain more access tokens.

**Relying Party (RP)**

Either a Liberty server configured as an OpenID Connect Client, or a client application that requires claims from an OpenID Provider (OP).

**Scope** Privilege or permission that is allowed to access resources of a third party.

**Token Endpoint**

A resource on an OP that accepts an authorization grant, or code, from a client in exchange for an access token, ID token, and refresh token.

**The Liberty server as an OpenID Connect Client**

You can configure the WebSphere Application Server Liberty to function as an OpenID Connect Client. This setup enables the Liberty server to rely on another Liberty server that is acting as an OP for user authentication and authorization.

A Liberty server that is configured to act as an OpenID Connect Client supports the Authorization Code Flow of the OpenID Connect 1.0 standard.

In the Authorization Code Flow, all token exchanges are handled by using the token endpoint of the OpenID Connect Provider. First, the client submits an authorization request to the authorization endpoint of the OP. Upon successful authentication and authorization with the OP, the client receives an authorization grant, or code, from the OP. This authorization code can then be sent in a request to the token endpoint of the OP. The client receives an ID token, an access token, and a refresh token in the response from the token endpoint. The client then validates the ID token and retrieves the subject identifier of the user. This profile flow is intended for clients that can securely maintain a client secret between themselves and the OP. This flow also allows clients to obtain a refresh token.

For configuring a Liberty server as an OpenID Connect Client, see “Configuring an OpenID Connect Client in Liberty” on page 1239

## The Liberty server as an OpenID Connect Provider

You can configure the WebSphere Application Server Liberty to function as an OpenID Connect Provider. This setup enables the Liberty server to act as an authorization server that can be used by OpenID Connect Clients.

A Liberty server that is configured to act as an OpenID Connect Provider supports the Authorization Code Flow and Implicit Flow of the OpenID Connect 1.0 standard. Each flow determines how the ID token, access token, and refresh token are returned to the client.

The Authorization Code Flow handles all token exchanges by using the token endpoint of the OpenID Connect Provider. An OpenID Connect Provider accepts an authorization request from a client at the OP's authorization endpoint. If authentication is necessary, the OpenID Connect Provider performs the appropriate authentication. The OpenID Connect Provider also obtains any required consent or authorization from the user, for example by prompting the user in a browser for permission to grant access to certain scopes. If successful, or if no authentication was required, the OpenID Provider sends back an authorization grant, or code, to the client. The OpenID Connect Provider then accepts a request that is submitted to its token endpoint by the client that includes the authorization code. The request that includes the authorization code is validated by the OpenID Provider. Upon successful validation, the OpenID Connect Provider returns a response to the client that includes an ID token and an access token.

In the Implicit Flow, unlike the Authorization Code Flow, all tokens are returned from the authorization endpoint; the token endpoint of the OP is not used. First, a client prepares and sends an authentication request to the authorization endpoint of the OP. The OpenID Connect Provider then performs any necessary authentication and also obtains any required consent or authorization from the user. For example, the OpenID Connect Provider prompts the user in a browser for permission to grant access to certain scopes. Upon successful authentication and authorization, the OpenID Connect Provider sends back an ID token and an access token to the client. The client then validates the ID token and retrieves the subject identifier of the user. This profile flow is intended for clients that cannot securely maintain a client secret between themselves and the OP, such as native applications.

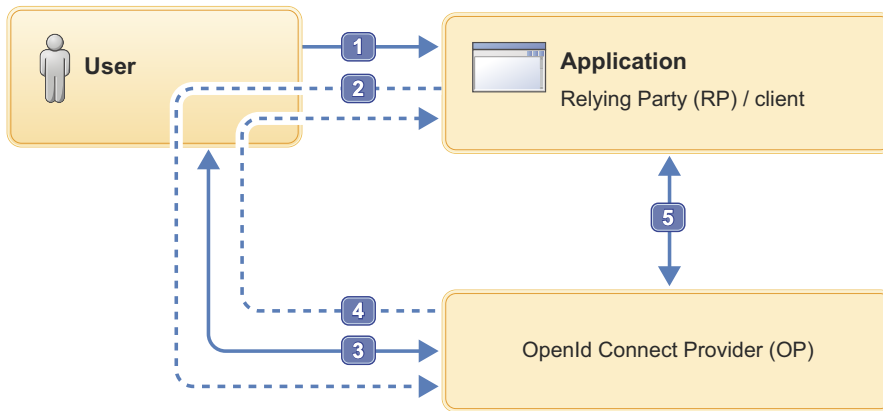
For configuring a Liberty server as an OpenID Connect Provider, see “Configuring an OpenID Connect Provider in Liberty” on page 1214

### Authorization Code Flow

A typical OpenID Connect Authorization Code Flow is described as follows:

1. A user accesses an application on the RP.
2. The RP prepares an authentication request and redirects the user to the OP.
3. The OP authenticates the user, for example by prompting the user for credentials. The user authorizes the RP to access the information that is required by the application. The OP generates a one-time use authorization code for the RP.
4. The OP redirects the user back to the RP with the authorization code.
5. The RP calls the token endpoint of the OP to exchange the authorization code for an access token, ID token, and a refresh token.
6. The RP uses the ID token to authorize the user.

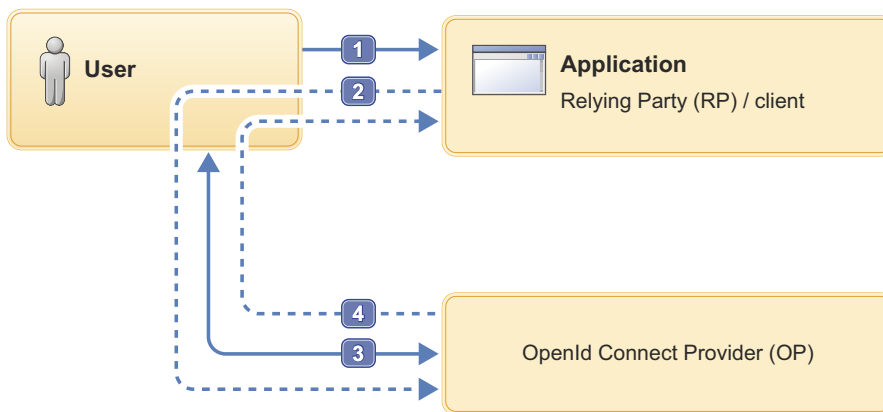




## Implicit Flow

The Implicit Flow is only supported by Liberty servers that are acting as an OpenID Connect Provider. A typical OpenID Connect Implicit Flow is described as follows:

1. A user accesses an application on the RP.
2. The RP prepares an authentication request and redirects the user to the OP.
3. The OP authenticates the user, for example by prompting the user for credentials. The user authorizes the RP to access the information that is required by the application.
4. The OP redirects the user back to the RP with an ID token and an access token.
5. The RP uses the ID token to authorize the user.



## Configuring an OpenID Relying Party in Liberty



8.5.5.4

You can configure a Liberty server to function as an OpenID Relying Party to take advantage of web single-sign-on.

### Before you begin

You must have at least one OpenID Provider (OP) that is trusted with authenticating users. Several third-party OpenID Providers are available.

## About this task

You can have users authenticated with an OpenID Provider by enabling the `openid-2.0` feature in Liberty, and in addition to other optional configuration information.

### Procedure

1. Add the `openid-2.0` Liberty feature to the `server.xml` file. Add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>openid-2.0</feature>
```

2.  Update the `server.xml` file with the OpenID Relying Party configuration options that are specified by an `<openId>` element.

You can either predefine an OpenID provider URL in your `server.xml` file by using the **`providerIdentifier`** attribute of the `<openId>` element, or you can package your application with `FormLogin` which gives users an option to submit an OpenID provider URL to use for authentication.

If the **`providerIdentifier`** attribute is added to the `server.xml` file, the Liberty server will automatically redirect users to the OpenID provider specified by that attribute. If the **`providerIdentifier`** attribute is not defined in the `server.xml` file, the Liberty server will first send a login form to ask the user to select or confirm an OpenID provider prior to redirecting the user to the OpenID provider.

The following is a sample OpenID configuration that defines an OpenID provider:

```
<openId id="myOpenId" providerIdentifier="https://openid.acme.com/op" userInfoRef="email">
 <userInfo id="email" alias="email" uriType="http://axschema.org/contact/email" count="1" required="true" />
</openId>
```


Adding the `openid-2.0` feature automatically enforces a certain minimum configuration. Consequently, there is no `<openId>` element that is required to be explicitly specified in the `server.xml` file. Without an `<openId>` element that is specified, the following configuration is implicit:

```
<openId id="myOpenId" userInfoRef="email">
 <userInfo id="email" alias="email" uriType="http://axschema.org/contact/email" count="1" required="true" />
</openId>
```

By default, the user's email address that is returned from the OpenID Provider is used for identity assertion and subject creation.

3. Configure the server's truststore to include the signer certificates of the OpenID Providers that are supported. For information about keystores, see [Enabling SSL communication for Liberty](#).
  - a. Extract the signer certificate from the OpenID Provider. Most major web browsers provide support for extracting or exporting certificates from websites through the browser interface.
  - b. Import the OpenID Provider certificate to the server's truststore. For one method of importing certificates into a truststore, see the `-import` flag capabilities of the **`keytool`** utility that is found in your Java installation directory.
  - c. Use the `sslRef` attribute of the `<openId>` element to point to your SSL configuration. If no `sslRef` attribute is specified, the default SSL configuration described in the keystore page mentioned previously will be used. Your SSL configuration should include the appropriate references to the truststore containing the imported OpenID Provider certificates.
4. Optional: Configure the Authentication Filter.

If the `providerIdentifier` attribute is configured inside the `openId` element in the `server.xml` file, you can configure `authFilterRef` to limit the requests that should be intercepted by the OpenID provider defined by the `providerIdentifier` attribute.

For more information on configuring the authentication filter, see  [8.5.5.5 "Authentication Filters"](#) on page 1252.

# Configuring SPNEGO authentication in Liberty



8.5.5.5

You can use single sign-on for HTTP requests by using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) web authentication for WebSphere Application Server Liberty. SPNEGO single sign-on enables HTTP users to log in to a Microsoft® domain controller only once at their desktop and to achieve single sign-on (SSO) with the Liberty server.

## Before you begin

Configure the following software and ensure that it is available:

1. A Microsoft Windows® Server running an Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC). For this topic, an example host for such a domain controller is `myAdMachine.example.com`. The domain controller name is `mydomain.example.com` and the Kerberos realm name is `MYDOMAIN.EXAMPLE.COM`, which is the domain controller name in all uppercase letters.
2. A Microsoft Windows® domain member (client) that supports the SPNEGO authentication mechanism as defined in IETF RFC 2478. Examples of an appropriate client might be a modern browser or a Microsoft .NET client. Most modern browsers support SPNEGO authentication. For this topic, an example host for the client is `myClientMachine.example.com`.
3. A server platform with a Liberty server that has a protected resource within an application. Users in the Active Directory must be able to access Liberty server protected resources by using a native Liberty server authentication mechanism. For this topic, an example Liberty server host is `myLibertyMachine.example.com`.

**Note:** The software configuration must have a running domain controller, at least one client machine in that domain and a server platform with a Liberty server that has a protected resource within an application, for a total of three required machines. Using SPNEGO directly from the domain controller is not supported.

**Note:** Ensure the clocks for the client, Microsoft Active Directory server, and Liberty server are synchronized to within 5 minutes of each other, by default. The allowable difference in synchronization is configurable.

**Note:** Only IBM JDKs are supported currently. Non-IBM JDKs are not supported.

## About this task

The objective of this task is to allow users to successfully access Liberty server resources without having to authenticate again, and thus achieve Microsoft Windows® desktop single sign-on capability.

This task demonstrates how to configure a Liberty server to support single sign-on for HTTP requests by using SPNEGO web authentication.

## Procedure

1. On the Microsoft domain controller (`myAdMachine.example.com`), create a Kerberos service principal name (SPN) and keytab file for the Liberty server. :
  - a. Create a user account for the Liberty server. This is the account that is used to map to the Kerberos service principal name (SPN). For Active Directory machines, this can typically be done by going to **Start > Administrative Tools > Active Directory Users and Computers**, right-clicking on **Users** in the panel, and selecting **New > User**. This topic assumes that the user `myLibertyMachine_http` was created with password security.

- b. Run the Microsoft **setspn** command to map the user account to a Kerberos SPN. This user account represents the Liberty server as being a Kerberos service with the KDC. The following is an example **setspn** command:

```
C:\> setspn -a HTTP/myLibertyMachine.example.com myLibertyMachine_http
```

```
Registering ServicePrincipalNames for CN=myLibertyMachine_http,CN=Users,DC=MYDOMAIN,DC=EXAMPLE,DC=COM
HTTP/myLibertyMachine.example.com
Updated object
```

**Note:** If your Microsoft **setspn** command version supports the **-S** option, then you must use the **-S** option instead of **-A**.

- c. Create the Kerberos keytab file by using the Microsoft **ktpass** tool. The default name for this file is **krb5.keytab**.

The following is an example **ktpass** command:

```
C:\> ktpass -out krb5.keytab -princ HTTP/myLibertyMachine.example.com@MYDOMAIN.EXAMPLE.COM -mapUser myLibertyMachine
```

```
Targeting domain controller: myAdMachine.MYDOMAIN.EXAMPLE.COM
Using legacy password setting method
Successfully mapped HTTP/myLibertyMachine.example.com to myLibertyMachine_http.
Key created.
Output keytab to krb5.keytab:
Keytab version: 0x502
keysize 93 HTTP/myLibertyMachine.example.com@MYDOMAIN.EXAMPLE.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x17 (
```

Make sure that there is not a duplicated SPN in the Microsoft forest by using one of the following commands:

- If your Microsoft **setspn** command version supports the **-X** option to search for a duplicated SPN, then use **setspn -X**:

```
C:\>setspn -X HTTP/myLibertyMachine.example.com
```

```
Processing entry 0
found 0 group of duplicate SPNs.
```

- You can also use the Microsoft **ldif** command. The following example shows that one entry was returned, meaning there is not a duplicated SPN.

```
C:\>ldifde -f check_SPN.txt -t 3268 -d "" -l servicePrincipalName -r "
(servicePrincipalName=HTTP/myLibertyMachine.example.com)" -p subtree
```

```
Connecting to "myAdMachine.MYDOMAIN.EXAMPLE.COM"
Logging in as current user using SSPI
Exporting directory to file check_SPN.txt
Searching for entries...
Writing out entries.
1 entries exported
```

For information on creating SPNs and keytab files on different KDC systems, see [Creating a Kerberos service principal name and keytab file](#).

2. On the Liberty server machine (**myLibertyMachine.example.com**), enable the Kerberos keytab and configuration files and SPNEGO web authentication.

- a. Copy the Kerberos keytab file from the domain controller to the Liberty server machine. The default name of this file is **krb5.keytab** and the default location varies depending on the platform but is the same directory as the Kerberos configuration file. For default locations for various platforms, see the next step.
- b. Create a Kerberos configuration file.

The Kerberos configuration file contains client configuration information, including the locations of KDCs for the realms of interest, defaults for the current Kerberos realm, and mappings of host names onto Kerberos realms. For Liberty servers, you must create this file manually.

The default location and name of this file varies depending on the operating system:

- **Windows** Windows - The default location is `c:\winnt\krb5.ini`. If the `krb5.ini` file is not in the `c:\winnt` directory, it might be in `c:\windows`
- **Linux** Linux - The default location is `/etc/krb5.conf`.
- **AIX** **HP-UX** **Solaris** AIX, z/OS, HP-UX, and Solaris - The default location is `/etc/krb5/krb5.conf`.

The following is a sample Kerberos configuration file for the AIX, z/OS, HP-UX, or Solaris platforms (based on the default keytab location):

```
[libdefaults]
 default_realm = MYDOMAIN.EXAMPLE.COM
 default_keytab_name = FILE:/etc/krb5/krb5.keytab
 default_tkt_enctypes = rc4-hmac
 default_tgs_enctypes = rc4-hmac
 forwardable = true
 renewable = true
 noaddresses = true
 clockskew = 300
 udp_preference_limit = 1

[realms]
 MYDOMAIN.EXAMPLE.COM = {
 kdc = myAdMachine.example.com:88
 default_domain = example.com
 }

[domain_realm]
 .example.com = MYDOMAIN.EXAMPLE.COM
```

**Note:** Realm names are usually specified in uppercase letters. If using Microsoft Active Directory, realm names are required to be uppercase.

**Note:** Not all of the KDC solutions available support all encryption types. Before choosing an encryption type, ensure that the KDC supports the encryption type that you want to use by consulting your Kerberos Administrator's and User's Guide.

Ensure that you have a common encryption type for the Kerberos configuration file, Kerberos keytab file, Kerberos SPN, and Kerberos client. For example, if the Kerberos client uses the RC4-HMAC encryption type, the target server must also support the RC4-HMAC encryption type and the Kerberos configuration file must list RC4-HMAC first in the **default\_tgt\_enctypes** and **default\_tkt\_enctypes** parameters.

For additional information and requirements on the content of this file, see *Creating a Kerberos configuration file*.

c. Verify the Kerberos configuration and keytab files.

- You can use the JDK command **klist** to list the SPN in the keytab file.
 

```
klist -k -t /etc/krb5.keytab
```
- You can use the JDK command **kinit** to validate the SPN in the keytab file and the Kerberos configuration file.
 

```
kinit -k -t /etc/krb5.keytab HTTP/myLibertyMachine.example.com
```

After the **kinit** command you can use the **klist** command to list the Kerberos ticket. If you get the Kerberos ticket, then the Kerberos keytab and configuration are valid.

d. Configure and enable SPNEGO web authentication on the Liberty server.

You can enable SPNEGO web authentication by enabling the `spnego-1.0` feature of the Liberty profile.

1) Add the `spnego-1.0` feature to the `server.xml` file.

```
<featureManager>
 <feature>spnego-1.0</feature>
 <feature>appSecurity-2.0</feature>
 ...
</featureManager>
```

Adding the spnego-1.0 feature automatically enforces a certain minimum configuration. You do not need to specify a <spnego> element in the server.xml file. Without specifying a <spnego> element, the following configuration is implicit.

```
<spnego
 canonicalHostName="true"
 disableFailOverToAppAuthType="true"
 trimKerberosRealmNameFromPrincipal="true"
 includeClientGSSCredentialInSubject="true" />
```

**Note:** The runtime forms the default SPN in the following format:

```
"HTTP/" + java.net.InetAddress.getLocalHost().getCanonicalHostName();
```

If the default SPN does not match what you have in the krb5.keytab file, then you need to specify the servicePrincipalNames, for example:

```
<spnego id="mySpnego" servicePrincipalNames="HTTP/myLibertyMachine.example.com"/>
```

**Note:** When the spnego-1.0 feature is enabled and the <spnego> element is either omitted or not configured with an authFilterRef attribute, all requests to access protected resources use SPNEGO authentication.

For more information on configuring the authentication filter, see  **8.5.5.5** "Authentication Filters" on page 1252.

**Note:** When values for the krb5Config or krb5Keytab attributes are not given, each respective file is expected to exist at its default location. The default locations for the Kerberos configuration and keytab files on various platforms are given earlier in this topic.

- Optional: Specify any additional configuration options as necessary. Liberty supports many common SPNEGO scenarios and configurations. For example, you can filter HTTP requests to require SPNEGO authentication for only certain requests, web applications, hosts, or user agents. Also, you can move the Kerberos configuration and keytab files away from their respective default locations. The following is a sample configuration that changes the default <spnego> settings:


```
<server>
 <featureManager>
 <feature>spnego-1.0</feature>
 <feature>appSecurity-2.0</feature>
 ...
 </featureManager>
 ...
 <authFilter id="myAuthFilter">
 <host id="myHost" name="example.com" matchType="contains" />
 <webApp id="myWebApp" name="protectedApp" matchType="equals" />
 </authFilter>

 <spnego id="mySpnego"
 includeClientGSSCredentialInSubject="false"
 krb5Config="${server.config.dir}/resources/security/kerberos/krb5.conf"
 krb5Keytab="${server.config.dir}/resources/security/kerberos/krb5.keytab"
 servicePrincipalNames="HTTP/myLibertyMachine.example.com"
 authFilterRef="myAuthFilter" />
 </spnego>
 ...
</server>
```

With this configuration, SPNEGO authentication is used for any requests that are received containing the host name example.com for resources within the web application protectedApp. In addition, the client's GSS credentials are not added to the user subject upon successful authentication. Finally, the Kerberos configuration and keytab files to be used by the server are given specific locations within the server configuration directory instead of their respective default locations.

For more configuration options, see  [8.5.5.5](#) The Simple and Protected GSS-API Negotiation Mechanism (SPNEGO).

For information on mapping Kerberos principal names to WebSphere user registry IDs, see Mapping of a client Kerberos principal name to the WebSphere user registry ID.

In the rare event that you wish to use Kerberos principal names for authorization, see  [8.5.5.5](#) “Using Kerberos principal name for authorization with SPNEGO authentication” on page 1202. These steps should be followed only when necessary and by users who specifically choose not to use the default mapping or JAAS custom login module mapping.

3. Configure the client application on the client application machine (`myClientMachine.example.com`).

The following steps must be done only on the client machine. Starting a browser on the Active Directory machine or Liberty server machine and performing these steps does not work.

The following steps are for users who are accessing SPNEGO-protected resources from a browser. You must have a browser installed that supports SPNEGO authentication.

Microsoft Internet Explorer:

- a. At the desktop, log in to the Windows Active Directory domain.
- b. In the Internet Explorer window, click **Tools > Internet Options**. In the window that is displayed, click the **Security** tab.
- c. Select the **Local intranet** icon and click **Sites**.
- d. If using Internet Explorer version 9 or older, go to the next step. If using Internet Explorer 10 or later, click **Advanced** in the Local intranet window.
- e. In the Local intranet window, complete the **Add this website to the zone** field with the web address of the host name so that single sign-on (SSO) can be enabled for the list of websites that are shown in the websites field. Your site information technology staff provides this information. Close the second Local intranet window and click **OK** to complete this step and close the Local intranet window.
- f. On the Internet Options window, click the **Advanced** tab and scroll to Security settings. Ensure that the **Enable Integrated Windows® Authentication** box is selected.
- g. Click **OK**. Restart your Microsoft Internet Explorer to activate this configuration.

Mozilla Firefox:

- a. At the desktop, log in to the Windows Active Directory domain.
- b. In the address field in Firefox, type `about:config`.
- c. In the Filter/Search box, type `network.n`.
- d. Double-click **network.negotiate-auth.trusted-uris**. This preference lists the sites that are permitted to engage in SPNEGO Authentication with the browser. Enter a comma-delimited list of trusted domains or URLs.

**Note:** You must set the value for `network.negotiate-auth.trusted-uris`.

- e. If the deployed SPNEGO solution is using the advanced Kerberos feature of Credential Delegation, double-click **network.negotiate-auth.delegation-uris**. This preference lists the sites for which the browser can delegate user authorization to the server. Enter a comma-delimited list of trusted domains or URLs.
- f. Click **OK**. The configuration reflects the updates.
- g. Restart your Firefox browser to activate this configuration.

**Note:** The user must be logged in to the domain controller for SPNEGO to work. Using the previous example machines, a user must log in to the domain controller at `MYDOMAIN.EXAMPLE.COM\username` in order for SPNEGO authentication through the browser to work.

**Note:** If you are prompted multiple times for a user ID and password, make sure that you enabled SPNEGO support on your client browser by following the previous instructions. You must also make sure that the `disableFailoverToAppAuthType` attribute in the `<spnego>` configuration is set to `false`.

## Results

Your Internet browser is now properly configured for SPNEGO authentication. You can use applications with secured resources that are deployed on Liberty servers without being prompted for a user ID and password.

To verify that SPNEGO is working, you can log in to the domain controller and then access a protected resource on the Liberty server, and because you are logged in to the domain controller, you are not prompted for credentials. However, if you do not log in to the domain controller and attempt to access a protected resource, you are prompted for credentials.

## Configuring Kerberos constrained delegation for out-bound SPNEGO tokens in Liberty

You can configure a Liberty server to support Kerberos constrained delegation for out-bound SPNEGO tokens.

### Before you begin

Make sure that you have configured SPNEGO web authentication.

Only IBM JDK 1.8 and later are supported.

### About this task

The Kerberos v5 extension called S4U (Services for Users) also known as constrained delegation comprises two parts:

#### S4U2self

Allows a Liberty server to obtain a service ticket to itself on behalf of a user. This can be used with any form of authentication that is supported by Liberty. `S4U2self` is the Kerberos Protocol Transition extension.

#### S4U2proxy

Allows a Liberty server to obtain service tickets to trusted services on behalf of a user. These service tickets are obtained by using the user's service ticket to the Liberty service. The services are constrained by the Kerberos Key Distribution Center (KDC) administrator. `S4U2proxy` is the Kerberos Constrained Delegation extension.

The constrained delegation feature provides the following APIs to create the out-bound SPNEGO token for back end services that support SPNEGO authentication, such as .NET servers and other Liberty servers.

- S4U2self API:  
`com.ibm.websphere.security.s4u2proxy.SpnegoHelper.buildS4U2proxyAuthorizationUsingS4U2self()`
- S4U2proxy API:  
`com.ibm.websphere.security.s4u2proxy.SpnegoHelper.buildS4U2proxyAuthorization()`

The following steps use the same example system setup that is used in [Configuring SPNEGO authentication in Liberty](#) and illustrated in [Single sign-on for HTTP requests using SPNEGO web authentication](#).



## Procedure

1. On the Microsoft domain controller `myAdMachine.example.com`, update the service principal name (SPN) that you use to validate the incoming SPNEGO token. For example, update the `HTTP/myLibertyMachine.example.com` SPN as follows:
  - a. To use `S4U2self`, perform the following steps:
    - 1) Open the user account that is mapped to the delegate SPN.
    - 2) Open the Attribute Editor tab.
    - 3) Modify the `userAccountControl` property as follows:
      - Trusted for auth delegation `0x1000000`, or the `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` enum) must be true.
    - 4) Set the trusted service:
      - Open the Delegation tab in the user account.
      - Select the **Trust this user for delegation to specified services only** radio button.
      - Select the **Use any authentication protocol** radio button.
      - Click on **Add** to add the trusted service.
      - Click **Users or Computers**.
      - Enter the SPN to be used for the trusted service.
      - Click **Check Names** and verify that the appropriate object name was found.
      - Click **OK**.
      - Select the **SPN specified** and click **OK**.
  - b. To use `S4U2proxy`, perform the following steps:
    - 1) Open the user account that is mapped to the delegate SPN.
    - 2) Open the Attribute Editor tab.
    - 3) Modify the `userAccountControl` property as follows:
      - Trusted for auth delegation (`0x1000000`, or the `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` enum) must be true.
    - 4) Set the trusted service:
      - Open the Delegation tab in the user account.
      - Select the **Trust this user for delegation to specified services only** radio button.
      - Click on **Add** to add the trusted service.
      - Click **Users or Computers**.
      - Enter the SPN to be used for the trusted service.
      - Click **Check Names** and verify that the appropriate object name was found.
      - Click **OK**.
      - Select the **SPN specified** and click **OK**.
2. On the Liberty server machine (`myLibertyMachine.example.com`), enable the constrained delegation feature by adding `constrainedDelegation-1.0` to the `featureManager` and configure the JAAS Kerberos login module in the `server.xml` file.

```
<featureManager>
 <feature>spnego-1.0</feature>
 <feature>constrainedDelegation-1.0</feature>
 ...
</featureManager>
<jaasLoginContextEntry id="com.ibm.security.jgss.krb5.accept" name="com.ibm.security.jgss.krb5.accept" loginModuleRef
 <jaasLoginModule id="useKeytab" className="com.ibm.security.auth.module.Krb5LoginModule" controlFlag="REQUIRED" lib
 <options
 credsType="both"
 debug="true"
 useDefaultCcache="false"
 tryFirstPass="true"
```

```

 forwardable = "true"
 principal = "HTTP/myLibertyMachine.example.com"
 useKeytab = "${server.config.dir}/resources/security/kerberos/krb5.keytab">
 </options>
</jaasLoginModule>

```

```

<library id="jaasSharedLib" apiTypeVisibility="spec, ibm-api, api">
 <fileset dir="${server.config.dir}/lib/global/" includes="*" />
</library>

```

When the `constrainedDelegation-1.0` feature is enabled, the following configuration is implicit:

```
<constrainedDelegation id="defaultConstrainedDelegation" s4U2selfEnabled="false" />
```

To use `S4U2self`, the following configuration is needed in the `server.xml` file:

```
<constrainedDelegation s4U2selfEnabled = "true" id="defaultConstrainedDelegation"/>
```

**Note:** When you use the `S4U2proxy` API, the `jaasLoginContextEntry` id and name `com.ibm.security.jgss.krb5.accept` can not change.

**Note:** By default, `S4U2proxy` is enabled and `S4U2self` is disabled. The `S4U2self` extension can be enabled or disabled by modifying the `s4U2selfEnabled` attribute in the `server.xml` file.

## Results

Your application is now ready to call the API provided by the constrained delegation feature.

## Using Kerberos principal name for authorization with SPNEGO authentication



8.5.5.5

You can use the fully qualified Kerberos principal name for authorization instead of using simple mapping or creating your own custom JAAS custom login module.

### About this task

The following steps should be rarely followed, and only by users who specifically choose not to use simple mapping, which is the default configuration, or choose not to add a JAAS custom login module to map the fully qualified Kerberos principal name to a user in the Liberty server user registry. This task allows you to use the fully qualified Kerberos principal name for authorization.

### Procedure

1. Configure the SPNEGO authentication to not trim the Kerberos realm name from the fully qualified Kerberos principal name by setting the `trimKerberosRealmNameFromPrincipal` attribute to `false`.
2. Configure the Liberty server to use either stand-alone LDAP or federated repositories.
 

For more information on how to configure LDAP, see [Configuring LDAP user registries with Liberty](#).

  - a. Make sure that the Active Directory user exists in the LDAP user registry and that this user has a single `userPrincipalName` attribute that is associated with it.
  - b. Update the LDAP filter in the `server.xml` file to search for the `userPrincipalName`, as shown in the following example:
 

```

<activatedLdapFilterProperties id="myactivatedfilters"
 userFilter="(&(userPrincipalName=%v))"
 groupFilter="(&(cn=%v))"
 userIdMap="*:userPrincipalName"
 groupIdMap="*:cn"
 groupMemberIdMap="ibm-allGroups:member">
</activatedLdapFilterProperties>

```
3. Configure the application bindings for the corresponding application to use the fully qualified Kerberos principal name as the user name along with a properly configured `access-id`. For example:

```

<application type="war" id="myApp" name="myApp" location="\${server.config.dir}/apps/myApp.war">
 <application-bnd>
 <security-role name="Employee">
 <user name="kevin@MYDOMAIN.EXAMPLE.COM" access-id="CN=kevin,CN=Users,DC=MYDOMAIN,DC=EXAMPLE,DC=COM"/>
 ...
 </security-role>
 ...
 </application-bnd>
</application>

```

## Customizing SSO configuration using LTPA cookies in Liberty

With single sign-on (SSO) configuration support, web users can authenticate once when accessing Liberty resources such as HTML, JavaServer Pages (JSP) files, and servlets, or accessing resources in multiple Liberty servers that share the same Lightweight Third Party Authentication (LTPA) keys.

### Example

When a user passes authentication on one of Liberty servers, authentication information generated by the server is transported to the web browser in a cookie. The cookie is used to propagate the authentication information to other Liberty servers.

The LTPA is configured and ready for immediate use. The default cookie name used to store the SSO token is called `ltpaToken2`. If you want to use a different name for the cookie, you can customize the cookie name using the **ssoCookieName** attribute of the `<webAppSecurity>` element. If you customize the cookie name, make sure that all the servers that participate in SSO use the same cookie name.

For more information about SSO, see SSO concept in Liberty.

The following example code sets the user to be logged out after the HTTP session expires and the name of the SSO cookie as `myCookieName`:

```
<webAppSecurity logoutOnHttpSessionExpire="true" ssoCookieName="myCookieName" />
```

**Note:** For SSO to work across Liberty servers, full profile servers, or both, set the following resources:

- The servers must use the same LTPA keys and share the same user registry.
- If the servers are not in the same domain, use the **ssoDomainNames** attribute of the `<webAppSecurity>` element to list the domains. The following example code sets the domain name to `domain.com`:
 

```
<webAppSecurity ssoDomainNames="domain.com" />
```
- If the servers are in the same domain, set the **ssoUseDomainFromURL** attribute of the `<webAppSecurity>` element to `true`, or specify the domain name in the **ssoDomainNames** attribute. The following example code sets **ssoUseDomainFromURL** to `true` so that the domain name is taken from the request URL:

```
<webAppSecurity ssoUseDomainFromURL="true" />
```

For details of all the available SSO settings, see the `<webAppSecurity>` element in **\*\*\*\* MISSING FILE \*\*\*\***.

## Configuring RunAs authentication in Liberty

You can delegate authentication to another identity by configuring the RunAs specification for Liberty.

### About this task

By mapping a specified user identity and optional password to a RunAs role, you can delegate the authentication process to a user that has the RunAs role.

You must enable the `appSecurity-2.0` and `servlet-3.0` Liberty features and have a user registry for your application to configure the RunAs role.

For more information about RunAs authentication, see “RunAs() authentication” on page 593.

To configure RunAs authentication, complete the following steps:

## Procedure

1. Enable the `appSecurity-2.0` and `servlet-3.0` Liberty features in the `server.xml` file.
2. Configure a user registry for your application.
3. Specify the `<run-as>` element in the deployment descriptor of your application.

The following example of `web.xml` file specifies subsequent calls be delegated to the user that is mapped to the role of `Employee`:

```
<servlet id="Servlet_1">
 <servlet-name>RunAsServlet</servlet-name>
 <display-name>RunAsServlet</display-name>
 <description>RunAsServlet</description>
 <servlet-class>web.RunAsServlet</servlet-class>
 <run-as>
 <role-name>Employee</role-name>
 </run-as>
</servlet>
```

4. **Distributed operating systems** Map the role that you specified in the previous step to a user. You can do this either in the `ibm-application-bnd.xml/xml` or in the `server.xml` file. In the `<run-as>` element, you must specify a user name. If you are using the `ibm-application-bnd.xml` file, the password is also required; if you are using the `server.xml` file, the password is optional. If the password is required, encode the password using the **securityUtility encode** command in the `/bin` directory. For more information about the **securityUtility** command, see “securityUtility command” on page 1162.

The following example uses the `<run-as>` element within the `<application-bnd>` element of the `server.xml` file, where the `Employee` role has been mapped to the RunAs user of `user5`:

```
<application-bnd>
 <security-role name="Employee">
 <user name="user1" />
 <user name="user5" />
 <run-as userid="user5" password="{xor}Lz4sLCgwLTs=" />
 </security-role>
</application-bnd>
```

### Note:

- Because the password is optional in the `server.xml` file, you can also use the following code for a user without a password:

```
<application-bnd>
 <security-role name="Employee">
 <user name="user1" />
 <user name="user5" />
 <run-as userid="user5" />
 </security-role>
</application-bnd>
```

- If you specify the `<application-bnd>` element in the `server.xml` file, your application must not be in the `dropins` folder. If you leave your application in the `dropins` folder, then you must disable application monitoring by setting the following in your `server.xml` file:

```
<applicationMonitor dropinsEnabled="false" />
```

The RunAs user name needs to be unique, and does not exist in external accounts. For example, if you authenticate a user to a SAML identity provider or OpenID Connect provider, make sure the RunAs user name is not in those external accounts.

For more information about the `run-as` element, see \*\*\*\* MISSING FILE \*\*\*\*.

## Configuring TAI in Liberty

You can configure Liberty to integrate with a third-party security service by using Trust Association Interceptors (TAI). The TAI can be called before or after single sign-on (SSO).

### Before you begin

Make sure that you have already installed a third-party security server as a reverse proxy server. The third-party security server can act as a front-end authentication server when the Liberty server applies its own authorization policy onto the resulting credentials, which are passed by the proxy server. You must also have a JAR file that contains the custom TAI class, which implements the `com.ibm.wsspi.security.tai.TrustAssociationInterceptor` interface.

**Note:** There is no support for monitoring changes of this JAR file.

### About this task

A TAI is used to validate HTTP requests between a third-party security server and a Liberty server. The TAI inspects the HTTP requests from the third-party security server to see whether they contain any security attributes. If the process of validating a request by the TAI is successful, the Liberty server authorizes the request by checking whether the client user has the required permission to access the resources.

For more information of custom TAI and SSO configuration with LTPA, see “Developing a custom TAI for Liberty” on page 1301 and “Customizing SSO configuration using LTPA cookies in Liberty” on page 1203.

You can also use the developer tools to configure a TAI service. **Distributed operating systems** For more information about the tools support, see “Configuring TAI on Liberty by using developer tools” on page 1206.

### Procedure

1. Enable the `appSecurity-2.0` Liberty feature in the `server.xml` file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
</featureManager>
```
2. Deploy your applications to the Liberty server and enable all Liberty features, such as, `jsp-2.2` and `jdbc-4.0`.
3. Place the TAI implementation library `simpleTAI.jar` in your server directory.
4. Update the `server.xml` file with the TAI configuration options and location of the TAI implementation library.

In the following `server.xml` file, the custom TAI is enabled, but does not do any authentication for unprotected URIs and does not allow to fallback to application authentication method if the TAI authentication fails. As shown in the example, the following configuration elements are available for TAI support:

- `trustAssociation`
- `interceptors`
- `properties`

```
<trustAssociation id="myTrustAssociation" invokeForUnprotectedURI="false"
 failOverToAppAuthType="false">
 <interceptors id="simpleTAI" enabled="true"
 className="com.sample.SimpleTAI"
 invokeBeforeSSO="true" invokeAfterSSO="false" libraryRef="simpleTAI">
 <properties prop1="value1" prop2="value2"/>
```

```

 </interceptors>
</trustAssociation>

<library id="simpleTAI">
 <fileset dir="{server.config.dir}" includes="simpleTAI.jar"/>
</library>
...

```

**Note:** The property name cannot start with a period (**.**), **config.**, or **service**. Also, the property name **id** or **ID** is not allowed.

**Note:** By default, the `invokeBeforeSSO` property is set to true. By using this setting TAI is invoked even when the SSO token is present and valid. However, if the expected behavior is to invoke TAI only when the SSO token is invalid or not present, then this property can be disabled by setting it to false, and enabling the `invokeAfterSSO` property. By using this setting TAI is invoked only when the SSO token is not present or is invalid. In some cases, this setup might improve the performance of your system.

For more information about the `<trustAssociation>`, `<interceptors>` and `<properties>` elements, see also **\*\*\*\* MISSING FILE \*\*\*\***.

## Configuring TAI on Liberty by using developer tools

### Distributed operating systems

You can configure a TAI service for Liberty using developer tools.

### Before you begin

For a description of the underlying process of configuring a server, and detailed information about specific aspects of server configuration, see “Administering Liberty manually” on page 946.

**Avoid trouble:** There are several security configuration examples on the `WASdev.net` website for reference when configuring security for your applications on Liberty.

### Procedure

1. Select the parent **Trust Association Interceptor** and enter an **ID** name.
2. Select the child **Trust Association Interceptor** and configure the **Class name** which is the fully qualified name of your TAI implementation class, then click the arrow next to the **Add** button and select **Global Element** to enter the shared library information.
3. Enter the **ID** for the shared library in the popup window and click **OK**.
4. Configure the **Name** and **Description** fields for the shared library, then click the arrow next to the **Add** button and select **Child Element** to add a fileset reference as a child element.
5. Configure the **Fileset**. Click **Browse** in the **Base directory** field and select the directory where the JAR file is located. Then, click **Browse** in the **Includes pattern** field to select your JAR file that contains your TAI implementation.
6. Configure **Interceptor properties** details by clicking **Add** to add properties for the interceptor.
7. Save the configuration. You can find the following configuration saved in the `server.xml` file.

```

<trustAssociation id="myTrustAssociation" invokeForUnprotectedURI="false"
 failOverToAppAuthType="false">
 <interceptors id="simpleTAI" enabled="true"
 className="com.ibm.websphere.security.sample.SimpleTAI"
 invokeBeforeSSO="true" invokeAfterSSO="false" libraryRef="simpleTAI">
 <properties hostName="machine1" application="test1"/>
 </interceptors>
</trustAssociation>

```

```
<library id="simpleTAI">
 <fileset dir="{server.config.dir}" includes="simpleTAI.jar"/>
</library>
```

## Configuring a custom form login page



8.5.5.4

Liberty provides the ability to define a custom form login page for users to submit authentication credentials.

### About this task

You can customize your own custom form login page, but you must implement this page in the required form-based authentication format as specified in the Servlet 3.0 specification. In all forms, the action on the form element must be `j_security_check`. The action must use the `j_username` input field to get the user name and the `j_password` input field to get the user password in forms supporting authentication schemes that require a user name and password. The custom form login page must be provided as an unprotected web resource. You can set this login page at the global server level, which applies to all applications deployed to the server. Alternatively, you can specify the login page for individual applications.

**Note:** Make sure that any files included in your form-login page (such as external style sheets, or images) are unprotected.

### Procedure

1. Specify the following form elements in the form login page that expects a user name and password.

```
<FORM action="j_security_check" method="POST">
 User name: <INPUT type="text" name="j_username">

 Password: <INPUT type="password" name="j_password">

 <INPUT type="submit" name="action" value="Login">
</FORM>
```

2. Configure the login form for use by applications on the server. There are two possible configurations to use a form login page in an application that is deployed to the server. You can configure the custom login page for use in a single application, or you can configure the page as a global login form that is used for all applications that are deployed to the server.
  - a. Configure a login form for a single application. You can configure individual applications to direct users to a specific form login page by configuring the `web.xml` file that is packaged with the application.

Specify the path to the login page in the `web.xml` file that is packaged with the application; for example:

```
<login-config>
 <auth-method>FORM/<auth-method>
 <realm-name>MyRealm/<realm-name>
 <form-login-config id="FormLoginConfig_1">
 <form-login-page>/login.jsp/<form-login-page>
 <form-error-page>/loginError.jsp/<form-error-page>
 </form-login-config>
</login-config>
```

To see how to customize and package a form login page, refer to [Customizing web application login](#).

- b. Configure a global login form. You can omit the form login page from the `web.xml` files that are packaged with each application and instead, specify in the `server.xml` configuration that the login form is for global use among applications that are deployed to the server.

In the `server.xml` file, include a `webAppSecurity` element with the `loginFormURL` attribute that is specified with a value of the path of the login form page; for example:

```
<webAppSecurity loginFormURL="myGlobalFormLogin/myLogin.jsp" />
```

Ensure that the form login page is packaged as a web application archive (WAR) file deployed to the server.

If the `form-login-page` element within an application's `web.xml` file does not exist, use the global login page that is specified in the server configuration.

3. **Optional:** Configure a custom form login page for OpenID.
4. **Optional:** Configure a custom form login page for OAuth.

## Configuring a custom form login page for OpenID



8.5.5.4

Login pages for Liberty must have a certain configuration in order to support OpenID Authentication.

### About this task



If you have configured the `providerIdentifier` attribute of the `<openId>` element in your `server.xml` file, you can skip this task.

Unlike standard form login pages that require a user name and password, form login for OpenID only requires submitting an OpenID Identifier. You must therefore configure form login pages to accept this OpenID Identifier.

### Procedure

1. Configure the login form to specify `j_security_check` as the value for the action.
2. Add an input field to the login form and set the `name` attribute to `openid_identifier` for users to input an OpenID Identifier. The resulting form is similar to the following HTML example:

```
<FORM action="j_security_check" method="POST">
 OpenID: <INPUT type="text" name="openid_identifier">

 <INPUT type="submit" name="action" value="Login">
</FORM>
```

No user name or password inputs are required if authentication is only performed using OpenID.

## Configuring a custom form login page for OAuth



8.5.5.4

You can configure custom form login pages for specific OAuth service providers.

### About this task

To use a custom form login page for a specific OAuth service provider, you must update the service provider definition in the `server.xml` file.

### Procedure

In the provider configuration, add the `customLoginURL` attribute and specify the login page URL as the value.

The following is an example custom login page entry in the provider definition:

```
<oauthProvider id="OAuthConfigSample"
 customLoginURL="https://acme.com:9043/oath20/login.jsp">
```

The login form that is used for OAuth must be configured to accept a user name and password.



# Configuring SAML Web Browser SSO in Liberty



8.5.5.7

You can configure a Liberty server to function as a SAML web browser Single-Sign-On (SSO) service provider.

## About this task

You can configure a Liberty server as a SAML web SSO service provider by enabling the `samlWeb-2.0` feature in Liberty, and in addition to other configuration information.

## Procedure

1. Add the `samlWeb-2.0` Liberty feature to your `server.xml` file by adding the following element declaration inside the `featureManager` element.

```
<feature>samlWeb-2.0</feature>
```

2. Liberty provides a default `samlWebSso20` element.

```
<samlWebSso20 id="defaultSP">
```

```
</samlWebSso20>
```

In this default configuration, the following default values are assumed:

- AssertionConsumerService URL:  
`https://<hostname>:<sslport> /ibm/saml20/defaultSP/acs`
- Service Provider (SP) metadata URL:  
`https://<hostname>:<sslport> /ibm/saml20/defaultSP/samlmetadata`

You can use a browser to download the metadata for this service provider (SP) by using this URL, and provide the URL to the SAML identity provider to establish federation between this SP and Identity Provider (IdP).

- The IdP metadata file must be copied to the `resources/security` directory on the server, and named `idpMetadata.xml`.
- The issuer name for SAML assertion is used as the security realm, and `NameID` is used as the principal to create an authenticated subject from the SAML assertion.
- The SAML `AuthnRequest` is signed with a private key in the default keystore of the server if the attribute `KeyStoreRef` is not specified. If the `keyAlias` is not configured, then `samlsp` is the default key alias. If the `keyAlias` is not configured, and the keystore only contains one private key, the private key is used in the signature.

**Note:** If you create a new service provider instance, and the `defaultSP` is no longer required, then you must explicitly disable the `defaultSP` instance, by adding the following to the `server.xml` file.

```
<samlWebSso20 id="defaultSP" enabled="false">
</samlWebSso20>
```

**Note:** It is required that you specify a URL safe String that is not null as the `id` for `samlWebSso20`. If the `id` is missing, then the configuration element is omitted, and is not handled as the `defaultSP`.

**Note:** When SAML is configured and enabled, all unauthenticated requests will use SAML authentication. To configure the types of requests that should and should not use SAML authentication, you must configure an authentication filter as described in step 15.

3. Optional: You can add `<samlWebSso20 id="defaultSP">` to the `server.xml` file, and customize the `defaultSP` service provider. For example:
  - `idpMetadata`: Add this parameter to change the IdP metadata location and file name from the default location and file name (`${server.config.dir}/resources/security/idpMetadata.xml`).

- `userIdentifier`: Add this parameter to select a SAML attribute name whose value is used as the principal name.
  - `groupIdentifier`: Add this parameter to select a SAML attribute name whose values are included as group members in the subject.
  - `realmName`: Use this parameter to explicitly specify the realm name to identify a SAML principal in this service provider. The default realm name is the SAML issuer name.
4. Optional: You can create one or more new `samlWebSso20` elements with a different ID. For example, if you create a new element with an ID as `mySP`, you effectively create a new SAML SP instance that has a new `AssertionConsumerService` URL:

`https://<hostname>:<sslport>/ibm/saml20/mySP/acs`

**Note:** The ID you choose for `samlWebSso20` is included in URL of the SP, including the `AssertionConsumerService` URL and metadata URL. The `samlWebSso20` ID must be non-empty and must not contain unsafe URL characters.

5. Optional: You can configure a trust engine. The Liberty SAML SP supports two types of trust engines:

- **Metadata trust engine:** Validates the signature against information that is provided in the configured IdP metadata.
- **PKIX trust engine:** Validates the trustworthiness of the certificate in the signature through PKIX validation. Certificates that pass this validation are assumed to be trusted.

Metadata is the default trust engine. If you want to use the PKIX trust engine, you need to add the `PKIXTrustEngine` element, and define the proper `trustAnchor`.

6. Optional: You can configure how to create an authenticated subject from SAML. By default the Liberty SP creates a subject from SAML assertion directly without the requirement of a local user registry, which is equivalent to the configuration `mapToUserRegistry=No`. The other configuration options are `mapToUserRegistry=User` or `mapToUserRegistry=Group`.

- `mapToUserRegistry=No`: The SAML issuer's name is `realm`, and the `NameID` is used to create a principal name and unique security name in the subject, and no group member is included. You can configure the attributes: `userIdentifier`, `realmIdentifier`, `groupIdentifier`, and `userUniqueIdentifier` to create an authenticated subject with a customized user name, realm name, group memberships, and unique security identifier.
- `mapToUserRegistry=User`: Choose this option if you want to validate a SAML user against your on-premises user registry, and create the user subject based on the on-premises registry.
- `mapToUserRegistry=Group`: Choose this option if you want to validate a SAML group against your local user registry, and create a subject to contain those validated groups. This option is similar to `mapToUserRegistry=No`, except for group memberships are verified against the on-premises user registry.

7. Optional: You can implement the Liberty SAML SPI, `com.ibm.wsspi.security.saml2.UserCredentialResolver` as a user feature to dynamically map a SAML assertion to a Liberty subject.
8. Optional: You can define rules to tell the IdP how to authenticate a user by configuring one or more of the following attributes when using an SP-initiated Web SSO flow: `forceAuthn`, `isPassive`, `allowCreate`, `authnContextClassRef`, and `authnContextComparisonType`.
9. Optional: You can define a required `NameID` format in the `AuthnRequest` by using the `nameIDFormat` attribute. You can specify any `NameID` format that is defined in the SAML specification, or use the keyword `customize` to specify the custom `NameID` format.
10. Optional: You can configure multiple SP and IdP federation partners by creating multiple `samlWebSso20` elements, and each `samlWebSso20` refers to one unique `authFilter` element. All `authFilters` must exclude each other. With multiple `samlWebSso20` configured, each can perform single-sign-on with its federated identity provider, and has its own authentication policy and consuming rules.

11. Optional: Add support for IdP-initiated unsolicited SSO. Liberty SAML SP supports IdP-initiated unsolicited SSO with and without the requirement of IdP metadata on-premises. If you do not have IdP metadata, or if you intend to use unsolicited SSO to federate with multiple identity providers with one Liberty SP, you must add the following configurations:
  - Configure <PKIXTrustEngine>, and import all the IdP signer certificates to the default truststore of the Liberty server, or to the trustAnchor of the PKIXTrustEngine.
  - Configure the trustedIssuers to list the issuer name of the IdP as it appears in the SAML assertion. The issuer name is used as the EntityID in the metadata.
  - If you intend to support unsolicited SSO only, you can configure SP-initiated unsolicited SSO as documented in the next step. This scenario is useful if the user's security context in the SP that is associated with SAML becomes invalid, the SP can redirect the user back to the IdP to start unsolicited SSO again automatically.
12. Optional: Add support for SP-initiated unsolicited SSO. The Liberty SAML SP uses configured IdP metadata to perform a solicited SAML AuthnRequest. It is possible for the Liberty SP to redirect unauthenticated requests to a preconfigured login application without using AuthnRequest. This scenario is useful if a business application performs pre-authentication processing before a user can authenticate to the SAML IdP, or the SAML IdP must be hidden from the Liberty SP. To configure this scenario, you add the loginPageURL attribute, and set its value to a URL that can instruct a user to authenticate to the SAML IdP.
13. Optional: Configure signature requirements with the following considerations:
  - SAML assertion. All SAML assertions must be digitally signed by the SAML IdP. In the rare case that you want to accept an unsigned assertion, you can explicitly configure wantAssertionsSigned=false.
  - The default signature algorithm is SHA256. If you must change the algorithm, use the signatureMethodAlgorithm attribute to modify it.
  - If you do not want to sign the SAML AuthnRequest, you can set authnRequestsSigned=false.
14. Optional: You can configure an SP authentication session and cookie. After SAML assertion is verified and processed, the Liberty SAML SP maintains an authenticated session between the browser and the SP without using an LTPA cookie. The authenticated session timeout is set to SessionNotOnOrAfter in the <saml:AuthnStatement> if presented, or to sessionNotOnOrAfter as configured in the server.xml file, with the default being 120 minutes. The session cookie name is automatically generated, and you can customize the cookie name by using the attribute spCookieName to specify the wanted name.

If you want the Liberty SP to create an LTPA cookie from the SAML assertion and use the LTPA cookie for subsequent authentication requests, you can add the configuration disableLtpaCookie=false. If you want to share the LTPA cookie with other servers, you must add the configuration attribute allowCustomCacheKey="false".

**Note:** If you configure disableLtpaCookie="false" and allowCustomCacheKey="false", ensure that a SAML user name is not directly authenticating to an on-premises user registry that prevents a user from having two accounts.

15. Optional: Configure the Authentication Filter.
 

When the samlWeb-2.0 feature is enabled, any unauthenticated request is authenticated through one SAML SP. If you define a customized samlWebSso20 element, all authentication requests are handled by this samlWebSso20 SP instance; otherwise, all authentication is handled by the default instance defaultSP. You can use authnFilter to define which SP instance to handle the authentication request.

For more information on configuring the authentication filter, see Authentication Filters.
16. Optional: Configure the SAML SP in a cluster.
 

If application servers are cluster members, and you use a router or reverse proxy server to route your requests, then you need to perform the following tasks:

  - The router and proxy server must be configured to support session affinity.

- Add the configuration attribute `spHostAndPort` to each application server, and set its value to the router or proxy server host name and port. For example, `spHostAndPort="https://myRouter.com:443"`.
- Generate an X509 certificate for signing the SAML AuthnRequest, and use this certificate on all application servers. For example, you can create a keystore to contain this certificate only, and add the `KeyStoreRef` to reference this keystore on all application servers.
- If `createSession="true"` is not set in a cluster environment, then the following error is encountered during the stress execution:

E CWWKS5029E: The relay state [sp\_initial\_KGe22fCWKG11D9VkoMuDz0Ji8pBxFPnU] in the response from the identity provider

Here is a sample cluster configuration:

```
<keyStore id="samlKeyStore" password="<password>"
 location="${server.config.dir}/resources/security/<samlKey.jks>" />

 <samlWebSso20 id="defaultSP"
 spHostAndPort="https://<IHS host>:<port>"
 keyStoreRef="samlKeyStore"
 createSession="true"
 allowCustomCacheKey="false"
 disableLtpaCookie="false"
 mapToUserRegistry="User">
 </samlWebSso20>
```

## Results

You have now established the configuration that is required to configure a Liberty server as a SAML service provider capable of single-signing on with SAML identity providers.

## Using OpenID Connect



8.5.5.4

OpenID Connect in the Liberty server is implemented as an OAuth 2.0 extension. In addition to providing OpenID Connect functions, the OpenID Connect provider supports all OAuth 2.0 functions.

### OpenID Connect endpoint URLs



8.5.5.4

Learn about OpenID Connect endpoint URLs that are available for communicating with the OpenID Connect provider.

After OpenID Connect is configured, several endpoint URLs are available on Liberty so that OpenID Connect clients can communicate with the OpenID Connect provider before accessing protected resources. By default, all communications must be over Transport Layer Security (TLS).

The following endpoint URLs are available for communicating with the OpenID Connect provider:

- “Authorization endpoint URL” on page 1213
- “Token endpoint URL” on page 1213
- “Introspection endpoint URL” on page 1213
- “UserInfo endpoint URL” on page 1213
- “Discovery endpoint URL” on page 1214
- “Coverage map endpoint URL” on page 1214
- “Registration endpoint URL” on page 1214

### Authorization endpoint URL

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/authorize`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  [8.5.5.4](#) Invoking the Authorization Endpoint for OpenID Connect.

### Token endpoint URL

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/token`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  [8.5.5.4](#) Invoking the Token Endpoint for OpenID Connect.

### Introspection endpoint URL

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/introspect`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  [8.5.5.4](#) Invoking the Introspection Endpoint for OpenID Connect.

### Userinfo endpoint URL

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/userinfo`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  **8.5.5.4** Invoking the UserInfo Endpoint for OpenID Connect.

**Discovery endpoint URL**

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/.well-known/openid-configuration`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  **8.5.5.4** Configuring an OpenID Connect Provider to accept discovery requests.

**Coverage map endpoint URL**

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/coverage_map`

where

**host\_name**

The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  **8.5.5.4** Invoking the coverage map service.

**Registration endpoint URL**

`https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/registration`

where

**host\_name**


The host name of the OpenID Connect provider.

**port\_number**

The secure port number that is configured on the Liberty server.

**provider\_name**

The OpenID Connect provider name.

For more information, see  **8.5.5.4** Configuring an OpenID Connect Provider to accept client registration requests.

**Configuring an OpenID Connect Provider in Liberty**

**8.5.5.4**

You can configure a Liberty server to function as an OpenID Connect Provider, or authorization server, to take advantage of web single sign-on.

## About this task

You can configure a Liberty server to act as an OpenID Connect Provider by enabling the `openidConnectServer-1.0` feature in Liberty, and in addition to other configuration information.

## Procedure

1. Add the `openidConnectServer-1.0` Liberty feature and any other needed features to the `server.xml` file. The `ssl-1.0` feature is also required for the `openidConnectServer-1.0` feature.

```
<feature>openidConnectServer-1.0</feature>
<feature>ssl-1.0</feature>
```

2. Define an OAuth service provider. OpenID Connect is built on top of the OAuth 2.0 protocol and you must configure a valid OAuth service provider. The configuration of an OAuth service provider includes the appropriate `oauth-roles`, `oauthProvider`, and `user registry` elements. Any user that is authorized to use OpenID Connect must also be mapped to the authenticated `oauth-role`. See [Defining an OAuth service provider](#) for more information.

The OAuth metadata is updated for OpenID Connect, and the main additions are in the client metadata. If you use the `databaseStore` mode for client registration, see “[Configuring an OpenID Connect Provider to accept client registration requests](#)” on page 1230 for more information. It is suggested that you follow the document to manage clients. If you use the `localStore` mode for client registration, you can register the `scope`, `preAuthorizedScope`, `grantTypes`, `responseTypes`, `introspectTokens`, and `functionalUserId`, as well as other attributes.

3. Add an `openidConnectProvider` element whose `oauthProviderRef` attribute references the configured `oauthProvider`. Each `oauthProvider` can only be referenced by one `openidConnectProvider`, and two or more `openidConnectProvider` elements cannot reference to the same `oauthProvider`. The `name` attribute and the `secret` attribute of the client element must match the `client ID` and the `client secret` of the corresponding OpenID Connect Client. This example works with the default Liberty server OpenID Connect Client.

**Note:** In this example, the OP expects the client's SSL port to be set to 443.

```
<openidConnectProvider id="OidcConfigSample" oauthProviderRef="OAuthConfigSample" />

<oauthProvider id="OAuthConfigSample">
 <localStore>
 <client name="client01" secret="{xor}LDo8LTor"
 displayName="client01"
 scope="openid profile email"
 redirect="https://server.example.com:443/oidcclient/redirect/client01"/>
 </localStore>
</oauthProvider>
```

**Note:** A valid client must register its name, redirect, scope, and secret for `authorization_code` grant type.

4. Configure the truststore of the server to include the signer certificates of the OpenID Connect Relying Parties, or clients, that are supported. For information about keystores, see “[Enabling SSL communication in Liberty](#)” on page 1152
5. Modify the SSL configuration of the server to use the configured truststore.

```
<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings" keyStoreRef="myKeyStore" trustStoreRef="myTrustStore" />
<keyStore id="myKeyStore" password="{xor}Lz4sLCgwLTs=" type="jks" location="{server.config.dir}/resources/security/B
<keyStore id="myTrustStore" password="{xor}Lz4sLCgwLTs=" type="jks" location="{server.config.dir}/resources/security
```

OpenID Connect is configured to use the default SSL configuration that is specified by the server. Therefore, the default SSL configuration for the server must use the truststore that is configured for OpenID Connect.

The user consent form in OpenID Connect is pluggable, which allows providers to create and maintain their own consent form. Because this form is retrieved over SSL, you must configure the truststore to include the signer certificate of the server on which the consent form is hosted. If the default consent form is used and the truststore that is used for OpenID Connect is configured to be different from the keystore that is used by the Liberty server, you must import the Liberty server's signer certificate into the OpenID Connect truststore.

For more OpenID Connect Provider configuration options, see \*\*\*\* MISSING FILE \*\*\*\*.



**Note:** In order to use OpenID Connect, the scope attribute must include `openid` in the scope list.

## Results


You have now completed the minimum configuration that is required to configure a Liberty server as an OpenID Connect Provider capable of communicating with other Liberty servers configured as OpenID Connect Clients.

### Using an OpenID Connect provider as an OAuth 2.0 authorization server: 8.5.5.4

An OpenID Connect provider can be used as a normal OAuth 2.0 authorization provider to issue an OAuth 2.0 `access_token`, and support all OAuth 2.0 grant types.

An OpenID Connect provider supports JSON Web Token (JWT) Bearer Token as a grant for requesting an OAuth 2.0 access token, see  JSON Web Token (JWT) for OAuth Client Authorization Grants and  Configuring an OpenID Connect Provider to accept JSON Web Tokens (JWT) for authorization grants.

If an authorization request is made with an authorization code grant or implicit grant type, and if `openid` scope is not included or approved, the request is handled as a normal OAuth authorization request. An `id_token` is not issued, and an `access_token` and `refresh_token` can be issued.

An OpenID Connect provider can support OAuth authorization flow with Resource Owner Password Credentials Grant or Client Credentials Grant, see  Configuring an OpenID Connect Provider to enable 2-legged OAuth requests.

### JSON Web Token (JWT) for OAuth Client Authorization Grants: 8.5.5.4

JWT for OAuth Client Authorization Grants enables a client to send a signed JWT token to the OpenID Connect Provider in exchange for an OAuth 2.0 access token.

JWT for OAuth Client Authorization Grants is included in the `openidConnectServer-1.0` feature. It enables a client to send a signed JWT token to the OpenID Connect Provider in exchange for an OAuth 2.0 access token.

An example usage scenario of this functionality might be a customer of an electric company who authorizes automatic monthly payments from an online bank. Assuming the electric company and the online bank have established a trusted relationship for the purposes of fulfilling such requests. The electric company can send a signed JWT Token with proper claims to the token endpoint URI of the OpenID Connect Provider that are configured for the online bank in order to request an OAuth 2.0 access token each month. The electric company can then use the access token to cash monthly payments from the online bank.

Portions of the JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants specification are supported for Liberty servers that are configured as OpenID Connect Providers. Users that want to support the JWT client functionality must do so by using their own application.



- Authorized scopes
- Claims in a JSON Web Token
- Submitting JSON Web Token requests

### Authorized scopes

The OpenID Connect Client sends an HTTPS request with a JWT to the token endpoint of the OpenID Connect Provider to request an access token. During this process, the user sees no consent form for authorizing the use of scopes. The JWT handler will handle the authorized scopes that are based on the following criteria:

1. If no scope parameter is specified in the request, the OpenID Connect Provider will not specify any scopes in the access token.
2. When an OpenID Connect Client is qualified as an autoAuthorized client in the OpenID Connect Provider configuration, any scope that is specified by the client in the request is specified in the scope list of the access token.
3. When an OpenID Connect Client is not qualified as an autoAuthorized client, the scope that is included in the request needs to be filtered by the list of scopes in the client configuration and must also be specified in the preAuthorizedScope list. If a scope in the HTTPS request is in the scope and preAuthorizeScope list of the client configuration, the scope can be specified in the scope list of the access token.

When a client is not qualified as an autoAuthorized client, scopes that can be included in the scope list of the access token must be properly configured in the client configuration. The scope must be included in the values for the scope and preAuthorizedScope attributes in the client configuration for the OpenID Connect Provider. In the example that is shown, the scopes profile and email are specified in the scope list of the access token because both are included in the scope and preAuthorizedScope value list. If a scope is not listed in the scope attribute of the client configuration, it is omitted from the scope list of the access token. If a scope is listed in the scope attribute but is not included in the preAuthorizedScope list within the client configuration, the authorization request triggers an invalid\_grant error in the response from the OpenID Connect Provider.

```
<openidConnectProvider id="OidcConfigSample" oauthProviderRef="OAuthConfigSample" />
<oauthProvider id="OAuthConfigSample" ... >
 ...
 <localStore>
 <client name="client01" secret="{xor}..."
 displayname="client01"
 scope="profile email phone"
 preAuthorizedScope="profile email"
 enabled="true"/>
 ...
 </localStore>
</oauthProvider>
```

### Claims in a JSON Web Token

A valid JSON Web Token must be signed. A Liberty server that is configured as an OpenID Connect Provider only supports HMAC-SHA256 as the token signing algorithm. The signing key for each OpenID Connect Client is the secret attribute in the client configuration of the OpenID Connect Provider. In the example that is shown, the signing key that is used would be "{xor}LD08LTor".

```
<client name="client01" displayname="client01" secret="{xor}LD08LTor" ... />
```

The OpenID Connect Provider also verifies the following claims in a JWT:

#### 'iss' (issuer)

This claim is required in a JWT. The iss claim must match the name attribute or the redirect

attribute of the client configuration in the OpenID Connect Provider. In the following example, the iss claim must match either client01 or http://op201406.ibm.com:8010/oauthclient/redirect.jsp.

```
<client name="client01" redirect="http://op201406.ibm.com:8010/oauthclient/redirect.jsp" scope="openid profile email">
```

#### 'sub' (subject)

This claim is required in a JWT. The value of the subject must be a valid user name in the user registry of the OpenID Connect Provider server.

#### 'aud' (audience)

This claim is required in a JWT. The value of the audience claim is the name of the issuerIdentifier when the issuerIdentifier attribute is specified in the openidConnectProvider configuration. If the issuerIdentifier attribute is not specified in the openidConnectProvider configuration, the audience must be the token endpoint URI of the OpenID Connect Provider. In the following example, the value of the audience claim would be "OpenIDConnectProviderID1".

```
<openidConnectProvider id="OidcConfigSample" oauthProviderRef="OAuthConfigSample" issuerIdentifier="OpenIDConnectProviderID1">
```

#### 'exp' (expiration)

This claim is required in a JWT and limits the time window that the JWT can be used. The OpenID Connect Provider verifies the exp against its system clock, plus some allowable clock skew.

#### 'nbf' (not before)

This is an optional claim. When present, the token is only valid after the time specified by this claim. The OpenID Connect Provider verifies this time against its system clock, plus some allowable clock skew.

#### 'iat' (issued at)

By default, this is an optional claim. However, if the iatRequired attribute of the jwtGrantType element is set to true, then all JWTs are required to contain the iat claim. When present, the iat claim indicates the time at which the JWT was issued. A JWT cannot be issued longer than the maxTokenLifetime.

#### 'jti' (JWT ID)

This is an optional claim and is the unique identifier of a JWT Token. When present, the same JWT ID cannot be reused by an issuer. For example, if client01 issues a JWT whose jti is id6098364921, then no other JWT issued by client01 can have a jti value of id6098364921. A JWT with a jti claim identical to another JWT is considered to be a replay attack. Liberty servers that are configured as OpenID Connect Providers set up a jti cache on the server. The size of the cache is specified by the maxJtiCacheSize in the jwtGrantType configuration. The jti IDs that are kept in the cache are checked against any new incoming jti ID. The jti IDs stored in the cache are not discarded unless the cache is full.

## Submitting JSON Web Token requests

It is a best practice to use the HTTPS protocol instead of HTTP to submit a JWT request. The token endpoint of the OpenID Connect Provider is used for handling HTTPS JWT requests. To determine the token endpoint for the OpenID Connect Provider, see Invoking the Token Endpoint for OpenID Connect or OAuth endpoint URLs.

The request must contain the following parameters:

- grant\_type - The value of this parameter must be "urn:ietf:params:oauth:grant-type:jwt-bearer"
- assertion - The value of this parameter must contain a single signed JWT Token.
- scope - This parameter is optional. If scope is omitted, the access token that is returned does not contain any scopes. The scope values listed in the scope parameter are checked against the OpenID Connect Provider configuration. For more information, see the previous Authorized Scopes section.
- client\_id - The value of this parameter must match the name attribute in the client configuration of the OpenID Connect Provider.



```

 try {
 signedAndSerializedString = jwtToken.serializeAndSign();
 } catch (SignatureException e) {
 throw e;
 }
 return signedAndSerializedString;
}
}

```

## Configuring an OpenID Connect Provider to accept discovery requests:



8.5.5.4

The discovery configuration endpoint makes information available about the capabilities that are supported by the OpenID Connect Provider (OP) server.

### About this task

The metadata that is returned by this service is based on and extends the OIDC Discovery 1.0 specification provider metadata. The service returns a set of default configurations if nothing is specified. Otherwise, refer to the list of properties to understand their purpose and possible configurable options.

### Procedure

You can override the default values for selected properties in the discovery configuration service. This action is performed by specifying the values in the `server.xml` file. Refer to the following table of properties to view the configurable properties and possible configuration options.

Table 87. Discovery request parameters

Attribute Name	Data Type	Required/Optional	Description
responseTypesSupported	Input	Optional	The response types that are supported by the OpenID Connect Provider (OP) server. Unless specified, the default values are code, token, and id_token token. More than 1 value can be specified. These values are strings. For example, possible values are: <ul style="list-style-type: none"> <li>code</li> <li>token</li> <li>id_token token</li> </ul>
subjectTypesSupported	Output only	N/A	The subject types that are supported by the OP server. This value is set to public. This value is a string.

Table 87. Discovery request parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
idTokenSigningAlgValuesSupported	Output only	Optional	<p>The ID token signing algorithms that are supported by the OP server. This value is specified as the server attribute <code>signatureAlgorithm</code> in the <code>openidConnectProvider</code> server configuration. Unless specified, the default value is HS256. Only 1 value can be specified. This is a string. For example, possible values for attribute <code>signatureAlgorithm</code> in the <code>openidConnectProvider</code> configuration:</p> <ul style="list-style-type: none"> <li>• none</li> <li>• RS256</li> <li>• HS256</li> </ul>
scopesSupported	Input	Optional	<p>The scope values supported by the OP server. Unless specified, the default values are <code>openid</code>, <code>general</code>, <code>profile</code>, <code>email</code>, <code>address</code>, and <code>phone</code>. More than 1 value can be specified. These values are strings. For example, possible values are:</p> <ul style="list-style-type: none"> <li>• <code>openid</code></li> <li>• <code>general</code></li> <li>• <code>profile</code></li> <li>• <code>email</code></li> <li>• <code>address</code></li> <li>• <code>phone</code></li> </ul>

Table 87. Discovery request parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
claimsSupported	Input	Optional	<p>The claims values that are supported by the OP server. Unless specified, the default values are sub, groupIds, name, preferred_username, picture, locale, email, and profile. More than 1 value can be specified. These values are strings. For example, possible values are:</p> <ul style="list-style-type: none"> <li>• sub</li> <li>• groupIds</li> <li>• name</li> <li>• preferred_username</li> <li>• picture</li> <li>• locale</li> <li>• email</li> <li>• profile</li> </ul>
responseModesSupported	Input	Optional	<p>The response modes that are supported by the OP server. Unless specified, the default values are query and fragment. More than 1 value can be specified. These values are strings.</p> <ul style="list-style-type: none"> <li>• query</li> <li>• fragment</li> </ul>
grantTypesSupported	Input	Optional	<p>The grant types that are supported by the OP server. Unless specified, the default values are authorization_code, implicit, refresh_token, client_credentials, password, and urn:ietf:params:oauth:grant-type:jwtbearer. More than 1 value can be specified. These values are strings. For example, possible values are:</p> <ul style="list-style-type: none"> <li>• authorization_code</li> <li>• implicit</li> <li>• refresh_token</li> <li>• client_credentials</li> <li>• password</li> <li>• urn:ietf:params:oauth:grant-type:jwtbearer</li> </ul>

Table 87. Discovery request parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
tokenEndpointAuthMethodsSupported	Supported	Optional	The token endpoint authorization methods that are supported by the OP server. Unless specified, the default values are <code>client_secret_post</code> , and <code>client_secret_basic</code> . More than 1 value can be specified. These values are strings. For example, possible values are: <ul style="list-style-type: none"> <li>• none</li> <li>• <code>client_secret_post</code></li> <li>• <code>client_secret_basic</code></li> </ul>
displayValuesSupported	Output only	N/A	The display values supported by the OP server. This value is set to page. This value is a string.
claimTypesSupported	Output only	N/A	The claim type values that are supported by the OP server. This value is set to normal. This value is a string.
claimsParameterSupported	Input	Optional	Indication of whether claims parameter is supported by the OP server. Unless specified, the default value is false. Only 1 value can be specified. This is a Boolean value. For example, possible values are: <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>
requestParameterSupported	Input	Optional	Indication of whether a request parameter is supported by the OP server. Unless specified, the default value is false. Only 1 value can be specified. This is a Boolean value. For example, possible values are: <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>

Table 87. Discovery request parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
requestUriParameterSupported	Input	Optional	Indication of whether request URI parameter is supported by the OP server. Unless specified, the default value is false. Only 1 value can be specified. This is a Boolean value. For example, possible values are: <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>
requireRequestUriRegistration	Input	Optional	Indication of whether require request URI registration is supported by the OP server. Unless specified, the default value is false. Only 1 value can be specified. This is a Boolean value. For example, possible values are: <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>

### Examples of discovery configuration

The following example assumes that the Liberty OP is configured with SSL on port 443.

`https://server.example.com:443/oidc/endpoint/<provider_name>/`

The discovery configuration endpoint is accessible at:

`https://server.example.com:443/oidc/endpoint/<provider_name>/.well-known/openid-configuration`

For example, in the `server.xml` file, a user can customize their OpenID Connect discovery configuration properties in the following manner:

```
<openidConnectProvider id="OidcConfigSample" oauthProviderRef="OAuthConfigSample">
 <discovery
 responseTypeSupported="token, id_token token"
 subjectTypesSupported="public"
 scopesSupported="openid, general, profile"
 claimsSupported="sub, groupIds, name"
 responseModesSupported="query"
 grantTypesSupported="implicit"
 tokenEndpointAuthMethodsSupported="client_secret_basic"
 displayValuesSupported="page"
 claimTypesSupported="normal"
 claimsParameterSupported="true"
 requestParameterSupported="true"
 requestUriParameterSupported="true"
 requireRequestUriRegistration="true"
 />
</openidConnectProvider>
<oauthProvider id="OAuthConfigSample">
</oauthProvider>
```



## Example of customized discovery configuration

Request Headers:

GET https://server.example.com:443/oidc/endpoint/<provider\_name>/.well-known/openid-configuration

Response Headers:

Status: 200

Content-Type: application/json

Cache-Control: public, max-age=3600

Response Body:

```
{
 "introspection_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/introspect",
 "coverage_map_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map",
 "issuer": "https://server.example.com:443/oidc/endpoint/<provider_name>",
 "authorization_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/authorize",
 "token_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/token",
 "response_types_supported": [
 "token",
 "id_token token"
],
 "subject_types_supported": [
 "public"
],
 "userinfo_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/userinfo",
 "registration_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/registration",
 "scopes_supported": [
 "openid",
 "general",
 "profile"
],
 "claims_supported": [
 "sub",
 "groupIds",
 "name"
],
 "response_modes_supported": [
 "query"
],
 "grant_types_supported": [
 "implicit"
],
 "token_endpoint_auth_methods_supported": [
 "client_secret_basic"
],
 "display_values_supported": [
 "page"
],
 "claim_types_supported": [
 "normal"
],
 "claims_parameter_supported": true,
 "request_parameter_supported": true,
 "request_uri_parameter_supported": true,
 "require_request_uri_registration": true,
 "check_session_iframe": "https://server.example.com:443/oidc/endpoint/<provider_name>/check_session_iframe",
 "end_session_endpoint": "https://server.example.com:443/oidc/endpoint/<provider_name>/end_session"
}
```

## Example of default discovery configuration

Request Headers:

GET https://server.example.com:443/oidc/endpoint/<provider\_name>/.well-known/openid-configuration

Response Headers:

Status: 200

Content-Type: application/json

Cache-Control:public, max-age=3600

Response Body:

```
{
 "introspection_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/introspect",
 "coverage_map_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map",
 "issuer":"https://server.example.com:443/oidc/endpoint/<provider_name>",
 "authorization_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/authorize",
 "token_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/token",
 "response_types_supported":[
 "code",
 "token",
 "id_token token"
],
 "subject_types_supported":[
 "public"
],
 "userinfo_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/userinfo",
 "registration_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/registration",
 "scopes_supported":[
 "openid",
 "general",
 "profile",
 "email",
 "address",
 "phone"
],
 "claims_supported":[
 "sub",
 "groupIds",
 "name",
 "preferred_username",
 "picture",
 "locale",
 "email",
 "profile"
],
 "response_modes_supported":[
 "query",
 "fragment"
],
 "grant_types_supported":[
 "authorization_code",
 "implicit",
 "refresh_token",
 "client_credentials",
 "password",
 "urn:iETF:params:oauth:grant-type:jwt-bearer"
],
 "token_endpoint_auth_methods_supported":[
 "client_secret_post",
 "client_secret_basic"
],
 "display_values_supported":[
 "page"
],
 "claim_types_supported":[
 "normal"
],
 "claims_parameter_supported":false,
 "request_parameter_supported":false,
 "request_uri_parameter_supported":false,
 "require_request_uri_registration":false,
 "check_session_iframe":"https://server.example.com:443/oidc/endpoint/<provider_name>/check_session_iframe",
 "end_session_endpoint":"https://server.example.com:443/oidc/endpoint/<provider_name>/end_session"
}
```

## Configuring claims returned by the UserInfo endpoint: 8.5.5.4

You can configure a Liberty OpenID Connect Provider to customize the claims that are returned by the UserInfo endpoint.

### About this task

You can configure the claims that are returned from a Liberty server OpenID Connect Provider by using the `scopeToClaimMap` and `claimToUserRegistryMap` subelements of the `openidConnectProvider` element in the `server.xml` file.

The OpenID Connect UserInfo endpoint accepts an access token as input and returns a set of claims about the user for whom the access token was created. The claims that are returned are determined by:

1. The scopes in the access token  
An access token can have multiple scopes. The scopes in an access token are the scopes that are supplied on the authorization endpoint invocation that created the access token.
2. The claims that are associated with the scopes  
Each scope can have multiple claims that are associated with it.
3. The federated repository properties that are associated with the claims  
A claim can have only one federated repository property that is associated with it.
4. The user registry attributes that are associated with the federated repository properties  
A federated repository property can have only one user registry attribute that is associated with it.

**Note:** The only user registry type that supports the retrieval of UserInfo claims is LDAP.

Liberty defines default scopes, claims, federated registry properties, and default mappings.

*Table 88. Default mappings for scopes, claims, and federated registry properties*

Scope	Claims	Federated registry property
profile	name, given_name, picture	displayName, givenName, photoURL
email	email	mail
address	address	postalAddress
phone	phone_number	telephoneNumber

Each of the following steps is optional. The Liberty server defines default scopes, claims, federated registry properties, and default mappings. The only time that you need to perform any of the following steps is if you want to change a default mapping or define a custom scope or claim.

### Procedure

1. Configure the claims that are associated with scopes. A scope can be mapped to multiple claims, and multiple claims must be comma-separated.

In the following example, the scope `CUSTOM_SCOPE1` is associated with two claims, `CUSTOM_CLAIM1` and `language`, and the scope `CUSTOM_SCOPE2` is associated with the claim `CUSTOM_CLAIM2`.

```
<scopeToClaimMap CUSTOM_SCOPE1="CUSTOM_CLAIM1, language"
 CUSTOM_SCOPE2="CUSTOM_CLAIM2" />
```

**Note:** Claim and scope names are case-sensitive, `CUSTOM_SCOPE1`, and `custom_scope1` are different scopes.

- a. To define scopes with the same spelling but different case, you must use the property subelement. In the following example, the scopes `CUSTOM_SCOPE1` and `custom_scope1` are defined.

```

<scopeToClaimMap CUSTOM_SCOPE1="CUSTOM_CLAIM1, language" >
 <property name="custom_scope1" value="custom_claim1,mobile"/>
</scopeToClaimMap>

```

2. Configure the federated repository properties that are associated with claims. A claim can be mapped to only one federated repository property.

In the following example, the claim CUSTOM\_CLAIM1 is associated with the federated repository property departmentNumber. The claim language is associated with the federated repository property preferredLanguage, and the claim CUSTOM\_CLAIM2 is associated with the federated repository property mail.

```

<claimToUserRegistryMap CUSTOM_CLAIM1="departmentNumber"
 language="preferredLanguage"
 CUSTOM_CLAIM2="mail" />

```

- a. To define claims with the same spelling but different case, you must use the property subelement. In the following example, the claims CUSTOM\_CLAIM1 and custom\_claim1 are defined.

```

<claimToUserRegistryMap CUSTOM_CLAIM1="departmentNumber" >
 <property name="custom_claim1" value="employeeType" />
</claimToUserRegistryMap>

```

3. Configure the user registry attributes that are associated with federated repository properties.

In the following example, the federated repository property photoURL is associated with the LDAP registry attribute ldapPersonPicture

```

<ldapRegistry...>
 ...
 <attributeConfiguration>
 <attribute name="ldapPersonPicture"
 propertyName="photoURL"
 entityType="PersonAccount" />
 </attributeConfiguration>
 ...
</ldapRegistry>

```

**Note:** The LDAP attribute must be defined in the schema of the LDAP registry.

## Results

You have now completed the configuration that is required to customize the claims that are returned by the UserInfo endpoint.

## Configuring an OpenID Connect Provider to enable 2-legged OAuth requests:

8.5.5.4

The typical OAuth flow consists of three “legs”, or stages of interaction between a client and an authorization server. In 2-legged OAuth scenarios, the client uses pre-authorized scopes so that no interaction with the user is necessary, removing the need to perform one of the legs in the typical flow. Specifically, the user does not need to authenticate to the authorization server or give consent for sharing the information that is specified by the requested scopes. Instead, all requested scope parameters are considered pre-authorized and are automatically added to the request token, which is then sent to the authorization server.

### Before you begin

This task expects you to have a Liberty server that is properly configured as an OpenID Connect Provider.

### About this task

In scenarios with two legs or stages, an Open ID Connect client can send a 2-legged HTTP request with a **grant type** of client\_credential or resource owner password. These requests do not go through the

authorization endpoint, so there is no scope consent form for users to confirm and approve the requested scopes; however the OpenID Connect Provider still needs to deal with the authorized scopes in its `access_token` content.

Liberty servers that are configured as OpenID Connect Providers that are equipped to handle 2-legged OAuth requests approve the pre-authorized scopes by using the following criteria:

1. If the **grant\_type** parameter value of a request is `client_credential` or `resource owner password` and the request is an OAuth 2.0 request, then all the scopes that are defined in the request are approved and are copied into the content of the access token. This is the existing behavior of the OAuth 2.0 feature.
2. If the request is an OpenID Connect request or a JWT Token OAuth, or OpenID Connect request, the following criteria is used:
  - If there is no scope parameter that is specified in the request, the OpenID Connect Provider will not accept the request.
  - The requested scopes must be present in the list of scopes that are defined by the scope attribute of the client configuration and must also be specified in the `preAuthorizedScope` list of the client configuration.

This task demonstrates how to configure a Liberty server that is acting as an OpenID Connect Provider to enable 2-legged OAuth requests.

## Procedure

To specify the list of pre-authorized scopes for a client, add the necessary scopes to the `scope` and `preAuthorizedScope` attributes of the client configuration within the appropriate `<oauthProvider>` element in your `server.xml` file. In the example that is shown, the scopes `profile` and `email` are qualified to be specified in the scope list of an access token that is returned by the OpenID Connect Provider. The `phone` scope is not considered a pre-authorized scope because it is missing from the `preAuthorizedScope` list.

```
<oauthProvider id="OAuthConfigSample" ...>

 <localStore>
 <client name="client01" secret="{xor}..."
 displayname="client01"
 scope="profile email phone"
 preAuthorizedScope="profile email"
 enabled="true"/>

 </localStore>
</oauthProvider>
```

**Note:** If a requested scope is not listed in the `scope` attribute of the client configuration, it is omitted from the scope of the access token that is returned. If a requested scope is listed in the `scope` attribute of the client configuration but is not included in the `preAuthorizedScope` list of the client configuration, it triggers an `invalid_grant` error in the response from the OpenID Connect Provider.

## Configuring an OpenID Connect Provider to use the RSA-SHA256 algorithm for signing of ID

tokens:  8.5.5.4

You can configure an OpenID Connect Provider to use the RS256 algorithm for the signing of ID tokens.

### About this task

You can configure an OpenID Connect Provider to use the RSA-SHA256 signature algorithm for signing ID tokens by setting the `signatureAlgorithm` to `RS256` and configuring a keystore with the private key used for signing.

## Procedure

1. In the `server.xml` file, create a `keystore` element that refers to the physical keystore that contains the private key that is capable of performing a RSA-SHA256 signature algorithm. For example:  

```
<keystore id="opTestKeyStore" location="{server.config.dir}/opKeyStore.jks" type="JKS" password="keystorePwd" />
```
2. Set the OpenID Connect Provider `signatureAlgorithm` attribute to `RS256`, set the `keyStoreRef` attribute to the `id` value of the keystore element that is used in step 1, and set the `keyAliasName` to locate the private key in the keystore. Setting the `keyStoreRef` is optional if the keystore element `id` used in step 1 is `opKeyStore`. For example:

```
<openidConnectProvider id="OAuthConfigSample" oauthProviderRef="OAuthConfigSample" signatureAlgorithm="RS256" keyStoreRef="opTestKeyStore" />
```

## Results

You have now configured an OpenID Connect Provider for signing ID tokens with RSA-SHA256.

## Configuring an OpenID Connect Provider to accept JSON Web Tokens (JWT) for authorization

grants:  8.5.5.4

You can configure a Liberty server that acts as an OpenID Connect Provider to accept a JSON Web Token in exchange for an access token.

### About this task

You can configure a Liberty server that acts as an OpenID Connect Provider to accept JSON Web Tokens by enabling the `openidConnectServer-1.0` and `ssl-1.0` features, in addition to other optional configuration information.

## Procedure

1. Ensure the `ssl-1.0` and `openidConnectServer-1.0` features are included in the feature manifest in the `server.xml` file.

```
<featureManager>
 <feature>ssl-1.0</feature>
 <feature>openidConnectServer-1.0</feature>
</featureManager>
```

2. Optional: Configure a `jwtGrantType` element inside of the appropriate `oauthProvider` element. The `jwtGrantType` element is optional. If no `jwtGrantType` element is included, the default values for all attributes are used; for example:

```
<oauthProvider id="OAuthConfigSample" ...>
 <jwtGrantType clockSkew="5m" iatRequired="false" tokenMaxLifetime="120m" maxJtiCacheSize="10000"/>
 ...
</oauthProvider>
```

For more JWT configuration options, see the section for the `jwtGrantType` element in the `**** MISSING FILE ****` topic.

## Configuring an OpenID Connect Provider to accept client registration requests: 8.5.5.4

The client registration endpoint is an administrator managed service that is used to register, update, delete, and retrieve information about an OpenID Connect Relying Party that intends to use the OpenID Connect Provider. In turn, the registration process can provide information for the Relying Party to use it, including the OAuth 2.0 Client ID and Client Secret, if not specified.

## Before you begin

The client registration service operates in one of two modes: local store or database store. These modes are determined by how the Liberty server configures its client store, whether clients are defined with the `oauthProvider localStore` attributes (local store) in the `server.xml` file or are configured with a database (database store).

In a local store configuration, the client registration service is limited to only retrieving OpenID Connect Relying Party information. You can modify the `server.xml` file to add more operations to register, update, or delete an OpenID Connect Relying Party.

In a database store configuration, there is no limitation on the client registration service and all operations are functional through the REST interface.

**Note:** A Liberty server must not configure its client store with both local store and database store. Choose only one configuration route.

The client registration endpoint is a protected administration endpoint with the `clientManager` role. To access this endpoint, the user must be granted the `clientManager` role by the administrator.

The `clientManager` role is one of the `oauth-roles` defined for an `oauthProvider`. The following is a sample configuration that shows a grant of the `clientManager` role to the user Alice or members in the `clientAdministrator` group.

```
<oauth-roles>
<authenticated>
<special-subject type="ALL_AUTHENTICATED_USERS" />
</authenticated>
<clientManager>
<group name="clientAdministrator" />
<user name="Alice" />
</clientManager>
</oauth-roles>
```

## About this task

Client registration information about an OpenID Connect Relying Party is largely used to define the usage scenario constraints of the client. Additionally, other operations of the OP that are opaque to the client use the client registration metadata to make authorization decisions.

The following example assumes that the Liberty OP is configured with SSL on port 443.

`https://server.example.com:443/oidc/endpoint/<provider_name>/registration`

The previous example also assumes that the `server.xml` file is configured with a user name: `clientAdmin` and password: `clientAdminPassword`, that uses the `oauth-role: clientManager`.

The client registration metadata consists of the following parameters:

*Table 89. Client registration parameters*

Attribute Name	Data Type	Required/Optional	Description
<code>client_id</code>	Input/Output	Optional	The client identifier that is being registered with the OP. Unless specified, this parameter value is generated during registration by default. This is a string.

Table 89. Client registration parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
client_secret	Input/Output	Optional	The client secret that is being registered with the OP. Unless specified, this parameter value is generated during registration by default. This is a string. During an update operation, the parameter value '*' preserves the existing value. A blank parameter value generates a new client_secret. A non-blank parameter value overrides the existing value with the newly specified value.
client_name	Input/Output	Optional	A description for the client that is being registered with the OP. Unless specified, this parameter is set to the client_id parameter value by default. This is a string.
application_type	Input	Optional	The application type that describes the client. Unless specified, the default value is web. This is a string. For example, possible values: <ul style="list-style-type: none"> <li>• &lt;an empty value is valid&gt;</li> <li>• web</li> <li>• native</li> </ul>
response_types	Input	Optional	The response type constraints that are used by this client. Unless specified, the default value is code. This is a JSON array. For example, possible values are: <ul style="list-style-type: none"> <li>• &lt;an empty value is valid&gt;</li> <li>• code</li> <li>• token</li> <li>• id_token token (order reversible)</li> </ul> <p>For a specific response_type, the corresponding grant_types must be specified. For more information, see response_types at the Client Metadata website.</p>



Table 89. Client registration parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
grant_types	Input	Optional	The grant type constraints that are used by this client. Unless specified, the default value is <code>authorization_code</code> . This is a JSON array. For example, the possible values are: <ul style="list-style-type: none"> <li>• &lt;an empty value is valid&gt;</li> <li>• <code>authorization_code</code></li> <li>• <code>implicit</code></li> <li>• <code>refresh_token</code></li> <li>• <code>client_credentials</code></li> <li>• <code>urn:iETF:params:oauth:grant-type:jwtbearer</code></li> <li>• <code>password</code></li> </ul>
redirect_uris	Input	Optional	The array of redirect URIs to which the client is constrained to. This is a JSON array.
post_logout_redirect_uris	Input	Optional	The array of post logout redirect URIs to which the client is constrained. This is a JSON array.
trusted_uri_prefixes	Input	Optional	The array of trusted URI prefixes that the client has deemed safe for sending access tokens. This is a JSON array.
scope	Input	Optional	The space delimited scope values to which the client is constrained. This is a string. If the client is allowed to request any scope, a value of <code>ALL_SCOPES</code> can be used.
preauthorized_scope	Input	Optional	The space delimited scope values that the client is preauthorizing that does not require user consent. This is a string.
subject_type	Input	Optional	The subject type constraint that is described by the client. This is a string. For example, possible values are: <ul style="list-style-type: none"> <li>• &lt;an empty value is valid&gt;</li> <li>• <code>public</code></li> </ul>

Table 89. Client registration parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
token_endpoint_auth_method	Input	Optional	<p>The token endpoint authentication method constraint that is used by the client. Unless specified, the default value is <code>client_secret_basic</code>. This is a string. For example, possible values are:</p> <ul style="list-style-type: none"> <li>• <code>&lt;an empty value is valid&gt;</code></li> <li>• <code>client_secret_basic</code></li> <li>• <code>client_secret_post</code></li> <li>• <code>none</code></li> </ul>
functional_user_id	Input	Optional	<p>This parameter indicates which user ID to associate with a request made on behalf of a client in a <code>client_credentials</code> grant type. This is a string.</p>
functional_user_groupIds	Input	Optional	<p>A list of group IDs to associate with access tokens obtained by this client using the client credentials grant type. The value is a list of group IDs to which the functional user is a member, where the group IDs are case-sensitive strings. The strings are defined by the authorization server. If the value contains multiple group IDs, their order does not matter. If the list is empty, the claim is omitted. When this client metadata parameter is specified, the value is returned in the <code>functional_user_groupIds</code> response parameter from the introspection endpoint for access tokens that are issued to this client with a client credentials grant. If the <code>functional_user_id</code> parameter is not used, this parameter is ignored.  <b>Note:</b> Authorization Servers must not trust the client to self-assert this parameter.</p>

Table 89. Client registration parameters (continued)

Attribute Name	Data Type	Required/Optional	Description
introspect_tokens	Input	Optional	A parameter value that indicates whether the specified client has the privilege to introspect an access token that is issued by the OP. This is a Boolean value.
registration_client_uri	Output only	N/A	A parameter that is returned in a response with the value that indicates the unique URL for a registered client. This is a string.
client_secret_expires_at	Output only	N/A	A parameter that is returned in a response with the value that indicates the number of seconds from 1970-01-01T0:0:0Z as measured in UTC, at which the client secret expires. The value 0 indicates no expiration time.
client_id_issued_at	Output only	N/A	A parameter that is returned in a response with the value that indicates the number of seconds from 1970-01-01T0:0:0Z as measured in UTC, at which the client ID was issued. The value 0 indicates that no client ID issue time was identified.

## Procedure

1. Register a client, as shown in the following example:

Request Headers:

```
POST https://server.example.com:443/oidc/endpoint/<provider_name>/registration
Accept: application/json
Content-Type: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

Request Payload:

```
{
 "token_endpoint_auth_method": "client_secret_basic",
 "scope": "openid profile email general",
 "grant_types": [
 "authorization_code",
 "client_credentials",
 "implicit",
 "refresh_token",
 "urn:iETF:params:oauth:grant-type:jwt-bearer"
],
 "response_types": [
 "code",
 "token",
 "id_token token"
],
}
```

```

"application_type":"web",
"subject_type":"public",
"post_logout_redirect_uris":[
 "https://server.example.com:9000/logout/",
 "https://server.example.com:9001/exit/"
],
"preauthorized_scope":"openid profile email general",
"introspect_tokens":true,
"trusted_uri_prefixes":[
 "https://server.example.com:9000/trusted/"
],
"redirect_uris":[
 "https://server.example.com:443/resource/redirect1",
 "https://server.example.com:9000/resource/redirect2"
]
}

```

#### Response Headers:

```

Status: 201
Cache-Control: private
ETag: "1B2M2Y8AsgTpgAmY7PhCfg=="
Content-Type: application/json

```

#### Response Body:

```

{
 "client_id_issued_at":1401776782,
 "registration_client_uri":"https://server.example.com:8020/oidc/endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret_expires_at":0,
 "token_endpoint_auth_method":"client_secret_basic",
 "scope":"openid profile email general",
 "grant_types":[
 "authorization_code",
 "client_credentials",
 "implicit",
 "refresh_token",
 "urn:iETF:params:oauth:grant-type:jwt-bearer"
],
 "response_types":[
 "code",
 "token",
 "id_token token"
],
 "application_type":"web",
 "subject_type":"public",
 "post_logout_redirect_uris":[
 "https://server.example.com:9000/logout/",
 "https://server.example.com:9001/exit/"
],
 "preauthorized_scope":"openid profile email general",
 "introspect_tokens":true,
 "trusted_uri_prefixes":[
 "https://server.example.com:9000/trusted/"
],
 "client_id":"b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret":"nmr0Q20CrMdw4pjqaImutZTcbQPzIoYgItjacCb9Wk33rKarhM3WDLmWIoE",
 "client_name":"b0a376ec4b694b67b6baeb0604a312d8",
 "redirect_uris":[
 "https://server.example.com:443/resource/redirect1",
 "https://server.example.com:9000/resource/redirect2"
]
}

```

## 2. Update a client, as shown in the following example:

#### Request Headers:

PUT https://server.example.com:443/oidc/endpoint/<provider\_name>/registration/registration/b0a376ec4b694b67b6baeb0604  
Accept: application/json  
Content-Type: application/json  
Authorization: Basic Y2xpZW50QRtaW46Y2xpZW50QRtaW5QYXNzd29yZA==

Request Payload:

```
{
 "token_endpoint_auth_method": "client_secret_basic",
 "scope": "openid profile",
 "grant_types": [
 "authorization_code"
],
 "response_types": [
 "code"
],
 "application_type": "native",
 "subject_type": "public",
 "post_logout_redirect_uris": [
 "https://server.example.com:9000/logout/"
],
 "preauthorized_scope": "openid",
 "introspect_tokens": false,
 "trusted_uri_prefixes": [
 "https://server.example.com:9003/trusted/"
],
 "client_id": "b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret": "*",
 "client_name": "updated client",
 "redirect_uris": [
 "https://server.example.com:443/resource/redirect1"
]
}
```

Response Headers:

Status: 200  
ETag: "3DD7affTGS91mfhPZ83B39Y=="  
Content-Type: application/json

Response Body:

```
{
 "client_id_issued_at": 1401776782,
 "registration_client_uri": "https://server.example.com:8020/oidc/endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret_expires_at": 0,
 "token_endpoint_auth_method": "client_secret_basic",
 "scope": "openid profile",
 "grant_types": [
 "authorization_code"
],
 "response_types": [
 "code"
],
 "application_type": "native",
 "subject_type": "public",
 "post_logout_redirect_uris": [
 "https://server.example.com:9000/logout/"
],
 "preauthorized_scope": "openid",
 "introspect_tokens": false,
 "trusted_uri_prefixes": [
 "https://server.example.com:9003/trusted/"
],
 "client_id": "b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret": "*",
 "client_name": "updated client",
}
```

```

 "redirect_uris":[
 "https://server.example.com:443/resource/redirect1"
]
 }
}

```

3. Retrieve a client, as shown in the following example:

Request Headers:

```

GET https://server.example.com:443/oidc/endpoint/<provider_name>/registration/registration/b0a376ec4b694b67b6baeb0604a3
Accept: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==

```

Response Headers:

```

Status: 200
Cache-Control: private
ETag: "3DD7affTGS91mfhPZ83B39Y=="
Content-Type: application/json

```

Response Body:

```

{
 "client_id_issued_at":1401776782,
 "registration_client_uri":"https://server.example.com:8020/oidc/endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a3",
 "client_secret_expires_at":0,
 "token_endpoint_auth_method":"client_secret_basic",
 "scope":"openid profile",
 "grant_types":[
 "authorization_code"
],
 "response_types":[
 "code"
],
 "application_type":"native",
 "subject_type":"public",
 "post_logout_redirect_uris":[
 "https://server.example.com:9000/logout/"
],
 "preauthorized_scope":"openid",
 "introspect_tokens":false,
 "trusted_uri_prefixes":[
 "https://server.example.com:9003/trusted/"
],
 "client_id":"b0a376ec4b694b67b6baeb0604a312d8",
 "client_secret":"*",
 "client_name":"updated client",
 "redirect_uris":[
 "https://server.example.com:443/resource/redirect1"
]
}

```

4. Retrieve a client (head request), as shown in the following example:

Request Headers:

```

HEAD https://server.example.com:443/oidc/endpoint/<provider_name>/registration/registration/b0a376ec4b694b67b6baeb0604a3
Accept: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==

```

Response Headers:

```

Status: 200
Cache-Control: private, no-cache=set-cookie
ETag: "3DD7affTGS91mfhPZ83B39Y=="
Content-Type: application/json

```

5. Delete a client, as shown in the following example:

Request Headers:

```

DELETE https://server.example.com:443/oidc/endpoint/<provider_name>/registration/registration/b0a376ec4b694b67b6baeb0604a3
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==

```

Response Headers:

Status: 204  
Content-Length: 0  
Content-Language: en-US

**Note:** The information in this topic also applies to client registration services of OAuth 2.0 clients, and OpenID Connect Relying parties.

### OpenID Connect custom forms: 8.5.5.4

You can replace the default form login page for user authentication, or develop your own user consent form to collect client authorization data.

### Authenticating a user: 8.5.5.4

OpenID Connect provider supports traditional Java Platform, Enterprise Edition (J2EE) FormLogin for user authentication.

You can customize the login form, see  8.5.5.4 OpenID Connect custom forms.

The OpenID Connect provider can be configured to support other authentication methods.

### OpenId Connect provider delegates user authentication to third-party authentication service

If you configure Trust Association Interceptor (TAI) to intercept a request to an OpenID Connect authorization endpoint (`/oidc/<provider name>/authorize`), the login form is not presented, and the user authentication is performed by the configured TAI.

### OpenId Connect provider authenticates user with HTTP Basic Authentication

If you want an OpenID Connect provider to authenticate a user with HTTP Basic Authentication, the openid connect authorization request must include a user id and password as defined in the Basic Authentication Scheme.

### OpenID Connect provider authenticates user with a client certificate

If you want an OpenID Connect provider to authenticate a user with a client certificate, you need to explicitly add the attribute `certAuthentication=true` inside the `oauthProvider` configuration element that is referenced by the `openidConnectProvider` configuration, and the user agent must be able to provide a client certificate for an OpenID Connect authorization request.

### OpenId Connect provider delegates user authentication to a third-party OpenId Connect provider

You can configure an OpenID Connect provider to delegate user authentication to a third-party OpenID Connect provider. To enable this authentication delegation, you configure the OP as an OpenID Connect relying party. Optionally, you can add an authentication filter to limit the `openidConnectClient-1.0` feature to protect an OpenID Connect authorization endpoint (`/oidc/<provider name>/authorize`) only.

## Configuring an OpenID Connect Client in Liberty

### 8.5.5.4

You can configure a Liberty server to function as an OpenID Connect Client, or Relying Party, to take advantage of web single sign-on and to use an OpenID Connect Provider as an identity provider.

## About this task

You can configure a Liberty server to act as an OpenID Connect Client by enabling the `openidConnectClient-1.0` feature in Liberty, and in addition to other configuration information.

## Procedure

1. Add the `openidConnectClient-1.0` Liberty feature and any other needed features to the `server.xml` file. The `ssl-1.0` feature is also required for the `openidConnectClient-1.0` feature. Add the following element declaration inside the `featureManager` element in your `server.xml` file:

```
<feature>openidConnectClient-1.0</feature>
<feature>ssl-1.0</feature>
```

2. Configure an `openidConnectClient` element. The following is an example of a minimal configuration that works with the default Liberty server OpenID Connect Provider.

The client must have a configured application available at the given URL pattern that can handle redirect requests from an OpenID Connect Provider. This URL must also precisely match the redirect URL registered for the client with the OP.

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443.

```
<openidConnectClient id="client01"
 clientId="client01"
 clientSecret="{xor}LDo8LTor"
 authorizationEndpointUrl="https://server.example.com:443/oidc/endpoint/OidcConfigSample/authorize"
 tokenEndpointUrl="https://server.example.com:443/oidc/endpoint/OidcConfigSample/token">
</openidConnectClient>
```

In this sample minimal configuration, the following default values are assumed:

- `scope=openid` profile: The scope of `openid` is required, and you can use the `scope` attribute to edit required scopes. For example, you can change the required scope to `openid profile email`.
  - This RP registers its redirect URL with the OP as `https://<host name>:<ssl port>/oidcclient/redirect/client01`, where both the host name and ssl port are automatically resolved, and `client01` is the ID of the `openidConnectClient` configuration element. If there is a proxy in front of the RP, you can override the host name and port with the attribute `redirectToRPHostAndPort`, and set `redirectToRPHostAndPort` to `https://<host name>:<ssl port>`.
3. Configure a user registry. User identities that are returned by the OP are not mapped to a registry user by default, so no users are required to be configured in the registry. However, if the `mapIdentityToRegistryUser` attribute of the `openidConnectClient` element is set to `true`, there must be a user entry for the appropriate identity that is returned from the OP in order for authentication and authorization to succeed. For more information about configuring a user registry, see “Configuring a user registry for Liberty” on page 1168.
  4. Configure the truststore of the server to include the signer certificates of the OpenID Connect Providers that are supported. For information about keystores, see “Enabling SSL communication in Liberty” on page 1152.
  5. Modify the SSL configuration of the server to use the configured truststore.

```
<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings" keyStoreRef="myKeyStore" trustStoreRef="myTrustStore" />
<keyStore id="myKeyStore" password="{xor}EzY90i0rJg==" type="jks" location="{server.config.dir}/resources/security/Ba
<keyStore id="myTrustStore" password="{xor}EzY90i0rJg==" type="jks" location="{server.config.dir}/resources/security/
```

OpenID Connect is configured to use the default SSL configuration that is specified by the server. Therefore, the default SSL configuration for the server must use the truststore that is configured for OpenID Connect.

6. **8.5.5.5** Optional: Configure a third party OpenID Connect provider.

To configure the Liberty OpenID Connect client to use a third party OpenID Connect Provider such as (Microsoft Azure or Google), you must configure the following attributes. The attribute values can be obtained by calling the OP's discovery endpoint, which provides a JSON document at the path that is formed by concatenating the string `/.well-known/openid-configuration` to the issuer.



- a. Set the `jwkEndpointUrl` attribute to the URL of the OP's JSON Web Key Set JWK document that is defined as `asjwks_uri` in the discovery file. For example, to use Google's OP, you can set `jwkEndpointUrl = "https://www.googleapis.com/oauth2/v2/certs"`.
- b. Set the `issuerIdentifier` attribute to the issuer as defined in the discovery file. An ID Token that does not contain this value as an `iss` claim is rejected. For example, you can set `issuerIdentifier="accounts.google.com"` if you are using Google as your OP.
- c. Set `signatureAlgorithm="RS256"`. The Liberty OpenID Connect client's default signature algorithm is HS256.
- d. Set the `userIdentityToCreateSubject` attribute to a claim name used by the vendor's ID Token that represents a user's unique identifier. For example, you can set `userIdentityToCreateSubject="email"` if you are using Google's OP, and `userIdentityToCreateSubject="upn"` or `userIdentityToCreateSubject="unique_name"` if you are using Microsoft Azure.
- e. Set the `groupIdentifier` attribute to the claim name that represents the user's group memberships or roles. For example, you can set `groupIdentifier="groups"` if you are using Microsoft Azure.

For more OpenID Connect Client configuration options, see \*\*\*\* MISSING FILE \*\*\*\*.

#### 7. Optional: Authentication Filter.

When the `openidConnectClient-1.0` feature is enabled and the `openidConnectClient` element is not configured with an `authFilterRef` attribute, any unauthenticated request is authenticated through the OpenID Connect provider.

For more information on configuring the authentication filter, see  [“Authentication Filters”](#) on page 1252.

#### 8. Support multiple OpenID Connect Providers.

You can configure Liberty as an OpenID Connect Relying Party to multiple OpenID Connect Providers by creating multiple `openidConnectClient` elements and multiple Authentication Filters. Each `openidConnectClient` element defines one Single-Sign-On relationship with one OpenID Connect Provider, and use the `authFilterRef` attribute to reference to one Authentication Filter.

#### 9. Configure a supported ID Token signature algorithm.

You can configure a Liberty OpenID Connect client to support the RS256 signature algorithm in an ID Token. The Liberty OpenID Connect client's default signature algorithm is HS256. If you configure RS256 as the ID Token's signature algorithm by setting `signatureAlgorithm="RS256"`, you must configure both the `trustStoreRef` and `trustAliasName`, unless the OP supports a JWK endpoint.

#### 10. Optional: Configure an “implicit” grant type.

The `openidConnectClient-1.0` feature uses an Authorization Code grant type to request a user authentication token, and you can configure the Liberty `openidConnectClient-1.0` feature to use an “implicit” grant type by adding `grantType="implicit"` to the `server.xml` file. If your Liberty server and OpenID Connect provider are in different firewalls, you must use this configuration option.

## Results

You have now established the minimum configuration that is required to configure a Liberty server as an OpenID Connect Client capable of communicating with other Liberty servers configured as OpenID Connect Providers.

### Invoking the Authorization Endpoint for OpenID Connect: [8.5.5.4](#)

In OpenID Connect the authorization endpoint handles authentication and authorization of a user.

## Before you begin

When starting the authorization endpoint from an in-browser client application or a client application implemented in a scripting language such as Javascript, for example, no configuration of a Liberty server as an OpenID Connect Client is necessary.

### About this task

The authorization endpoint accepts an authentication request that includes parameters that are defined by both the OAuth 2.0 and OpenID Connect 1.0 specifications.

In the Authorization Code Flow, the authorization endpoint is used for authentication and authorization and returns an authorization grant to the client. This authorization grant can then be passed in a request by the client to the token endpoint in exchange for an ID token, access token, and refresh token. In the Implicit Flow, the authorization endpoint still performs authentication and authorization but also directly returns an ID token and access token to the client in its response; no interaction is performed with the token endpoint.

A Liberty server with OpenID Connect enabled has access to the OpenID Connect authorization endpoint at the following URL:

```
https://server.example.com:443/oidc/endpoint/<provider_name>/authorize
```

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443.

### Procedure

1. Prepare an HTTP GET or POST request that includes the following required and recommended parameters.
  - **scope:** (Required) OpenID Connect requests must contain the `openid` scope value. Other scopes may also be present
  - **response\_type:** (Required) Determines the authorization processing flow to be used. When using the Authorization Code Flow, this value is `code`. When using the Implicit Flow, this value is `id_token` or `id_token access_token`. No access token is returned when the value is `id_token`
  - **client\_id:** (Required) Client identifier that is valid at the OpenID Connect Provider.
  - **redirect\_uri:** (Required) Redirection URI to which the response will be sent. This value must exactly match one of the redirection URI values for the registered client at the OP.
  - **state:** (Recommended) Opaque value used to maintain state between the request and the callback.
  - **nonce:** (Required for the Implicit Flow) String value used to associate a client session with an ID token and to mitigate replay attacks.

You can include more parameters in the request. For a description of other supported parameters, see the OpenID Connect Core 1.0 specification.

We do not support only an `id_token` `response_type`. Using the implicit flow must always use `id_token` and will return an access token.

2. Send the GET or POST request to the authorization endpoint URL.

### Results

After completing these steps, you have a valid HTTP request that is being sent to the authorization endpoint. The authorization endpoint returns a response in the manner described in the Examples section.

The OpenID Connect Provider attempts to authenticate and authorize the user once it receives a request from the client.

In the Authorization Code Flow, if authentication and authorization succeed, the OpenID Connect Provider issues an authorization code and includes it as a parameter in an OAuth 2.0 Authorization Response to the client. If the initial request included **state**, the authorization response will also include the exact **state** value that was included in the initial request. Using the application/x-www-form-urlencoded format, the **code** and **state** parameters are added as query parameters to the **redirect\_uri** value that was specified in the authorization request.

In the Implicit Flow, if authentication and authorization succeed, the following parameters are returned from the authorization endpoint.

- **access\_token**: Access token. This is returned unless the **[response\_type]** value in the initial request is **[id\_token]**.
- **token\_type**: OAuth 2.0 Token Type. For OpenID Connect, this value is Bearer.
- **id\_token**: ID token.
- **state**: Required if included in authorization request.
- **expires\_in**: (Optional) Expiration time of the access token in seconds since the response was generated.

These parameters are added to the fragment component of the **redirect\_uri** value that is specified in the authorization request, not as query parameters such as in the Authorization Code Flow.

## Example

The following examples show forms of Authorization and Implicit code flow.

An example request for the Authorization Code Flow is shown here:

```
GET /authorize?
 response_type=code
 &scope=openid profile email
 &client_id=client01
 &state=af0ifjsldkj
 &redirect_uri=https://server.example.com:443/oidcclient/redirect/client01 HTTP/1.1
```

An example request for the Implicit Flow is shown here:

```
GET /authorize?
 response_type=id_token token
 &scope=openid profile
 &client_id=client01
 &state=af0ifjsldkj
 &redirect_uri=https://server.example.com:443/oidcclient/redirect/client01
 &nonce=n-0S6_wzA2Mj HTTP/1.1
```

An example response from the authorization endpoint in the Authorization Code Flow is shown here:

```
HTTP/1.1 302 Found
Location: https://server.example.com:443/oidcclient/redirect/client01
 code=Sp1x10BeZQQYbYS6WxSbIA
 &state=af0ifjsldkj
```

An example response from the authorization endpoint in the Implicit Flow is shown here:

```
HTTP/1.1 302 Found
Location: https://server.example.com:443/oidcclient/redirect/client01
 access_token=S1AV32hkKG
 &token_type=Bearer
 &id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso
 &expires_in=3600
 &state=af0ifjsldkj
```

## Invoking the Token Endpoint for OpenID Connect: 8.5.5.4

In the OpenID Connect Authorization Code Flow, the token endpoint is used by a client to obtain an ID token, access token, and refresh token.

### Before you begin

When starting the token endpoint from an in-browser client application or a client application implemented in a scripting language such as Javascript, for example, no configuration of a Liberty server as an OpenID Connect Client is necessary.

### About this task

The token endpoint accepts a request from the client that includes an authorization code that is issued to the client by the authorization endpoint. When the authorization code is validated, the appropriate tokens are returned in a response to the client.

The token endpoint is not used in the OpenID Connect Implicit Flow.

A Liberty server with OpenID Connect enabled has access to the OpenID Connect token endpoint at the following URL:

```
https://server.example.com:443/oidc/endpoint/<provider_name>/token
```

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443. All communication with the token endpoint must use TLS.

### Procedure

1. Prepare an HTTP POST request with the following parameters.
  - **grant\_type:** The value of this parameter must be `authorization_code`.
  - **code:** The authorization code received from the authorization endpoint.The parameters must be added by using the `application/x-www-form-urlencoded` format.
2. POST the request to the token endpoint URL.

### Results

After completing these steps you have a valid HTTP POST request that is being sent to the token endpoint. The token endpoint returns a response as described in the Examples section.

When the OpenID Connect Provider validates the token request that is received from the client, the OpenID Connect Provider returns an HTTP 200 response back to the client with a JSON object in `application/json` format. The response includes the ID token, access token, and refresh token, along with the following additional parameters:

- **token\_type:** OAuth 2.0 Token Type. For OpenID Connect, this value is `Bearer`.
- **expires\_in:** Expiration time of the access token in seconds since the response was generated.

All responses from the token endpoint that contain tokens, secrets, or other sensitive information have their **Cache-Control** header value set to **no-store** and **Pragma** header value set to **no-cache**.

.

## Example

The following shows examples of an HTTP POST request and response

An example request is shown here:

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
 grant_type=authorization_code
 &code=Sp1x10BeZQQYbYS6WxSbIA
 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

An example response is shown here:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "access_token": "S1AV32hkKG",
 "token_type": "Bearer",
 "refresh_token": "8xL0xBtZp8",
 "expires_in": 3600,
 "id_token": "eyJ ... zcifQ.ewo ... NzAKfQ.ggw8h ... Mzqg"
}
```

## Invoking the Introspection Endpoint for OpenID Connect:

8.5.5.4

The introspection endpoint enables holders of access tokens to request a set of metadata about an access token from the OpenID Connect Provider that issued the access token. The access token must be one that was obtained through OpenID Connect or OAuth authentication.

### Before you begin

When a resource service or a client application invokes the introspection endpoint, it must register itself as a normal OAuth 2.0 client to the OpenID Connect server. The registered client metadata must include the attribute `introspectTokens = true`.

### About this task

Information that is contained within access tokens that are used in OpenID Connect and OAuth 2.0 is opaque to clients. This can enable protected resources or clients to make authorization decisions that are based on the metadata that is returned from the OpenID Connect Provider about the access token.

A Liberty server with OpenID Connect enabled has access to the OpenID Connect introspection endpoint at the following URL:

```
https://server.example.com:443/oidc/endpoint/<provider_name>/introspect
```

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443.

### Procedure

1. Set up client authentication with the client ID and password for a registered OpenID Connect Client in the HTTP Basic Authorization header of a GET or POST request. The client ID and password are encoded by using the `application/x-www-form-urlencoded` encoding algorithm. The encoded client ID is used as the username and the encoded password is used as the password.
2. Include the string value for the access token as a parameter in the GET or POST request to the introspection endpoint.
3. Send the GET or POST request to the introspection endpoint URL.

## Results

After completing these steps you have a valid HTTP request that is being sent to the introspection endpoint as shown in the Examples section.

For valid requests, the introspection endpoint returns an HTTP 200 response with a JSON object in application/json format that includes the following information, depending upon whether the access token is active or expired.

When the access token is active, the endpoint returns active:true, as well as the following additional information in the JSON object:

**active** Boolean indicator of whether the access token is active.

**client\_id**

Client identifier of the OpenID Connect Client who requested the access token.

**sub** Resource owner who authorized the access token.

**scope** Space-separated list of scopes that are associated with the access token.

**iat** Integer timestamp, measured in seconds since January 1, 1970 UTC, indicating when the access token was issued.

**exp** Integer timestamp, measured in seconds since January 1, 1970 UTC, indicating when the access token will expire.

**realmName**

Realm name of the resource owner.

**uniqueSecurityName**

Unique security name of the resource owner.

**tokenType**

Access token type. For OpenID Connect, this value is Bearer.

**grant\_type**

String indicating the type of grant that generated the access token. Possible values are: authorization\_code, password, refresh\_token, client\_credentials, resource\_owner, implicit, and urn:iETF:params:oauth:grant-type:jwt-bearer.

If the access token is expired, but the provided authentication is valid, or if the provided access token is of the wrong type, the endpoint returns active:false in the JSON object.

**Note:** For a client or resource service to perform access token introspection, the client or resource service must register itself as a client to the OpenID Connect provider, and the client metadata must have introspect\_tokens set to true.

## Example

The following shows examples of an active and expired access token along with a request.

An example request is shown here:

```
POST /register HTTP/1.1
Accept: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbU13
token=SOYleDziTitHeKcodp6vqEmRwKPjz31FZTcsQtVC
```

An example response for an active access token:

```

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
 "exp" : 1415307710,
 "realmName" : "BasicRealm",
 "sub" : "testuser",
 "scope" : "openid scope2 scope1",
 "grant_type" : "authorization_code",
 "uniqueSecurityName" : "testuser",
 "active" : true,
 "token_type" : "Bearer",
 "client_id" : "pclient01",
 "iat" : 1415307700
}

```

Example response for an expired access token:

```

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
 "active":"false"
}

```

## Invoking the coverage map service: 8.5.5.4

The coverage map service is an unprotected endpoint that returns a JavaScript Object Notation (JSON) array of slash-terminated URI prefixes. The array of URI prefixes designates which web contexts are part of a Single Sign On (SSO) group, thus enabling clients to know whether a URI destination is deemed safe to send an access token.

### Before you begin

The coverage map service returns a JSON array of URI prefixes, which are a unique set that is derived from the aggregation of the **trusted\_uri\_prefixes** parameter values that are specified in the registered clients. Therefore, a typical case for populating the coverage map service is to register clients and specify the **trusted\_uri\_prefixes** value.

### About this task

The following example assumes that the Liberty OpenID Connect provider is configured with SSL on port 443.

```
https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map
```

Additionally, this example assumes that a client is registered with the specified **trusted\_uri\_prefixes**.

### Procedure

1. Specify a **token\_type URI** query parameter on the coverage\_map endpoint. The following is an example request that assumes that the client is registered with the specified trusted\_uri\_prefixes  

```
https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map?token_type=bearer
```

The only token\_type value that is supported is **token\_type=bearer**.
2. Get the coverage map for the bearer token type, as shown in the following example.  
Request Headers:  

```
GET https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map?token_type=bearer
```

Response Headers:

```
Status: 200
CacheControl: public, maxage=600
ETag:"vvhkgXkRx+BzR3Q4kwCCqw=="
ContentType: application/json

Response Body:
[
 "http://res1.ibm.com/",
 "https://trusted.server.ibm.com:9554/resources/"
]
```

## Invoking the UserInfo Endpoint for OpenID Connect:



8.5.5.4

The UserInfo endpoint returns claims about a user that is authenticated with OpenID Connect authentication.

### About this task

To obtain the claims for a user, a client makes a request to the UserInfo endpoint by using an access token as the credential. The access token must be one that was obtained through OpenID Connect authentication. The claims for the user who is represented by the access token are returned as a JSON object that contains a collection of name-value pairs for the claims. The UserInfo endpoint is an OAuth 2.0 protected resource, which means that the credential required to access the endpoint is the access token.

The claims that are returned by the UserInfo endpoint can be customized with the OpenID Connect Provider configuration, see [Configuring claims returned by the UserInfo endpoint](#).

A Liberty profile server with OpenID Connect enabled has access to the OpenID Connect UserInfo endpoint at the following URL:

```
https://server.example.com:443/oidc/endpoint/<provider_name>/userinfo
```

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443.

### Procedure

1. Set up authentication with an access token that was obtained through OpenID Connect authentication. The access token can be provided in the HTTP Basic Authorization header or with the `access_token` request parameter. In either case, the access token does not need to be encoded.
2. Send the GET or POST request to the UserInfo endpoint URL.

### Results

After completing these steps you have a valid HTTP request that is being sent to the UserInfo endpoint as shown in the Examples section.

For valid requests, the UserInfo endpoint returns an HTTP 200 response with a JSON object in `application/json` format that includes the claims that are configured for the OpenID Connect Provider.

### Example

The following examples illustrate requests with a valid token and invalid tokens.

- Request that uses the HTTP Bearer Authorization header to pass the access token
- Response for a valid access token
- Invalid access tokens

An example request that uses the HTTP Bearer Authorization header to pass the access token:



```
POST /register HTTP/1.1
Accept: application/x-www-form-urlencoded
Authorization: Bearer fAAAdL01c6QWDbPs9HrWHz5e7nRWWAnxqTTP7i88G
```

The token can also be passed by using the `access_token` request parameter:

```
POST /register HTTP/1.1
Accept: application/x-www-form-urlencoded
 access_token=fAAAdL01c6QWDbPs9HrWHz5e7nRWWAnxqTTP7i88G
```

It is a best practice to use the HTTP Authorization header instead of the `access_token` request parameter because HTTP request parameters, which can include sensitive information, can be saved in the browser history or cache.

Here is an example response for a valid access token. The `sub` and `groupIds` claims are always returned. The other claims that are shown here are the default claims for an OpenID Connect Provider.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "sub" : "bob",
 "groupIds" : ["bobsdepartment", "administrators"],
 "given_name" : "Bob",
 "name" : "Bob Smith",
 "email" : "bob@mycompany.com",
 "phone_number" : "+1 (604) 555-1234;ext5678",
 "address" : { "formatted" : "123 Main St., Anytown, TX 77777" },
 "picture" : "http://mycompany.com/bob_photo.jpg"
}
```

For an invalid access token, the `UserInfo` endpoint returns an HTTP 401 status code with an error message in the `WWW-AUTHENTICATE` header.

```
HTTP/1.1 401 Unauthorized
CONTENT-LENGTH : 0
WWW-AUTHENTICATE : Bearer error=invalid_token,
 error_description=CWwKS1617E: A userinfo request was made with
 an access token that was not recognized. The request URI was
 /oidc/endpoint/MyOAuthProvider/userinfo.
```

## Invoking the Session Management Endpoint for OpenID Connect:

8.5.5.4

The session management endpoint enables OpenID Connect Relying Parties to monitor the login status of a user with a particular OpenID Connect Provider (OP) while minimizing network traffic. With the help of the session management endpoint, a Relying Party (RP) can log out a user who logged out of the OpenID Connect Provider.

### Before you begin

The OP session management endpoint URL is obtained from the `check_session_iframe` attribute in the discovery information that is returned from the discovery endpoint of the OP. This URL must be used as the target of an `iframe` in the RP application that requires session management functionality. The RP application must also know the `id` attribute of the `iframe` in order to submit `Window.postMessage()` requests to it.

### About this task

To help determine the login status of a user, the RP loads an `iframe` with its `src` target set to the session management endpoint of the OP. The session management endpoint has access to a cookie that stores the login status, or browser state, of a user. This browser state cookie is updated when a user logs out of the

OP. The RP can then use client side scripting to invoke the `Window.postMessage()` function of the OP iframe, sending the client ID and the currently known session state in the text of the message. If the RP receives a `postMessage` back from the OP frame with a value of `changed`, then the login status of the user at the OP has changed and the RP can decide whether to log out the user. If a value of `unchanged` is returned, the user is still logged in at the OP.

A Liberty server with OpenID Connect enabled has access to the OpenID Connect session management endpoint at the following URL:

```
https://server.example.com:443/oidc/endpoint/<provider_name>/check_session_iframe
```

**Avoid trouble:** In this example, the client expects the SSL port to be set to 443.

### Procedure

1. Create a web resource in an appropriate RP application capable of loading an iframe that targets the OP session management endpoint. The web resource also needs to have access to the session state value that is returned in the `session_state` parameter of the authorization response. The session state value can be stored in a cookie, for example, or in any other way that allows client-side scripting in the web resource to know what the value is. The following is a sample HTML snippet for such an iframe.

```
<iframe id="iframeOP" src="https://server.example.com:443/oidc/endpoint/0idcConfigSample/check_session_iframe" frameborder="0">
```

2. To check the login status of a user, invoke the `Window.postMessage()` function of the OP iframe, passing the client ID and session state as the message parameter in the format `Client ID + " " + Session State` and the host name of the OP as the target origin parameter. In the following JavaScript sample function, the script expects the session state value to be stored in a cookie that is named `session_state`, and the `getCookieValue()` function returns the value that is stored in the `session_state` cookie.

```
var targetOP = "https://server.example.com:443";
function checkStatus() {
 var client = "client01";
 var sessionState = getCookieValue("session_state");
 var text = client + " " + sessionState;
 var iframe = document.getElementById("iframeOP");
 iframe.contentWindow.postMessage(text, targetOP);
}
```

3. Configure the web resource to listen for `postMessages` from the OP that contain the value `changed` or `unchanged` to reflect the respective login status of the user. The RP can then decide whether to log the user out of the RP based on the value that is returned from the OP. The function must ensure that the origin of the `postMessage` matches the expected OP host name. Any messages that do not match are rejected. The following JavaScript example shows how to add an event listener to the web resource to listen for such messages.

```
var targetOP = "https://server.example.com:443";
window.addEventListener("message", receiveMessage, false);
function receiveMessage(event) {
 if (event.origin !== targetOP) {
 // Origin did not come from the OP; this message must be rejected.
 return;
 }
 if (event.data === "unchanged") {
 // User is still logged in to the OP
 } else {
 // User has logged out of the OP
 }
}
```

### Results

You now have a web resource on the RP that is capable of utilizing the session management functionality of OpenID Connect on a Liberty server OP. The browser state that is maintained in the OP iframe is

updated when users log in or log out of the OP. After successful login at the OP, a new session state value is provided in the authorization response to the RP. The RP can then use client-side scripting to validate the session state of the user to determine whether the login status of the user changed on the OP without broadcasting extra network traffic.

## Example

The following HTML example shows a complete HTML page that uses OpenID Connect Session Management. The src attribute of the OP iframe is set to the session management endpoint URL obtained from the OP. The startChecking() function is automatically called every 60 seconds and checks the login status of the user. The page has a message event listener that calls the receiveMessage() function when a postMessage is received. This function makes sure the postMessage comes from the expected domain for the OP and checks the value of the returned message to see whether the login status of the user is changed or unchanged.

You can load this HTML page, itself, as an invisible iframe in another web resource within the RP. This enables any web resource that loads this iframe to monitor the login status of the user on the client side.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>iFrame RP Page</title>
</head>
<body onload="javascript:startChecking()">
 <iframe id="iframeOP" src="https://localhost:8999/oidc/endpoint/OidcConfigSample/check_session_iframe" frame
</body>
<script>
 var targetOP = "https://server.example.com:443";

 window.addEventListener("message", receiveMessage, false);

 function startChecking() {
 checkStatus();
 // Check status every 60 seconds
 setInterval("checkStatus()", 1000*60);
 }

 function getCookieValue(cookieName) {
 var name = cookieName + "=";
 var cookies = document.cookie.split(';');
 if (!cookies) {
 return null;
 }
 for (var i = 0; i < cookies.length; i++) {
 var cookie = cookies[i].trim();
 if (cookie.indexOf(name) == 0) {
 return cookie.substring(name.length, cookie.length);
 }
 }
 return null;
 }

 function checkStatus() {
 var client = "client01";
 var sessionState = getCookieValue("session_state");
 var text = client + " " + sessionState;
 var iframe = document.getElementById("iframeOP");
 iframe.contentWindow.postMessage(text, targetOP);
 }

 function receiveMessage(event) {
 if (event.origin !== targetOP) {
 // Origin did not come from the OP; this message must be rejected
 }
 }
</script>
</body>
</html>
```

```

 return;
 }
 if (event.data === "unchanged") {
 // User is still logged in to the OP
 } else {
 // User has logged out of the OP
 }
}
</script>
</html>

```

## Authentication Filters



8.5.5.5

You can use the authentication filter to determine whether certain HTTP servlet requests are processed by certain providers.

Liberty server authentication filter uses the filter criteria that are specified in the `authFilter` element in the `server.xml` file to determine whether certain HTTP servlet requests are processed by certain providers, such as OpenID, OpenID Connect, or SPNEGO, for authentication.

If all conditions in the `authFilter` element are met, the HTTP servlet request is processed by the particular provider that references that `authFilter` element. If any of the conditions within the `authFilter` element are not met, the HTTP servlet request is not processed by the provider.

### Supported elements

The `authFilter` element supports the following elements: `userAgent`, `host`, `webApp`, `remoteAddress`, and `requestUrl`.

- The `userAgent` element is compared against a corresponding header value that is extracted from the incoming HTTP servlet request. The `userAgent` element is compared against the “User-Agent” HTTP request header, which identifies the client software that is used by the originating request. For web client browsers, this value reflects the browser type that is used to initiate the request (Internet Explorer, Firefox, Safari, etc.).
- The `host` element is used similarly to the `userAgent` element. The `host` element is compared against the “Host” HTTP request header, which identifies the target host name of the request.
- The `webApp` element is used to specify the application, or list of applications, hosted on the Liberty server that is protected by this authentication filter.
- The `remoteAddress` element is compared against the remote TCP/IP address of the client application that sent the HTTP request. You can configure wildcards for specifying subnets and ranges by using the `lessThan` or `greaterThan` values of the `matchType` attribute, as shown among the examples that follow later in this topic.
- The `requestUrl` element is compared against the URL that is used by the client application to make the request. Single URL patterns are configured or piped lists of values are configured, as shown among the examples that follow later in this topic.

### Authentication Filter examples

#### Request URL contains a pattern

The following example shows a typical configuration for an authentication filter. Here, any incoming request with a request URL containing `/SimpleServlet` is processed by the service that is configured to use this filter.

```

<authFilter id="myAuthFilter">
 <requestUrl id="myRequestUrl" urlPattern="/SimpleServlet" matchType="contains"/>
</authFilter>

```

### Request URL contains one of a set of patterns

In the following example, a piped list of request URL patterns is specified. To process an incoming request with the service configured to use this filter, the incoming request URL must contain any one of `/SimpleServlet`, `/EmployeeRoleServlet`, or `/AllRoleServlet`.

```
<authFilter id="myAuthFilter">
 <requestUrl id="myURL" urlPattern="/SimpleServlet|/EmployeeRoleServlet|/AllRoleServlet" matchType="contains"/>
</authFilter>
```

### Web application name contains a pattern

In the following example, a web application name is specified in the authentication filter. Incoming requests must target the `"myApp"` application to be processed by the service that is configured to use this filter.

```
<authFilter id="myAuthFilter">
 <webApp id="myWebApp" name="myApp" matchType="contains"/>
</authFilter>
```

### Web application name contains one of a set of patterns

In the following example, a piped list of web applications is specified. To process an incoming request with the service configured to use this filter, the incoming request must target any one of the `"myApp1"`, `"myApp2"`, or `"myApp3"` applications.

```
<authFilter id="myAuthFilter">
 <webApp id="myWebApp" name="myApp1|myApp2|myApp3" matchType="contains"/>
</authFilter>
```

### Request originates from a certain IP address

The following example shows how to use wildcards in the `remoteAddress` element. With this configuration, the service that is configured to use this filter processes the incoming request if the request comes from an IP address anywhere in the `127.0.0.*` range.

```
<authFilter id="myAuthFilter">
 <remoteAddress id="myRemoteAddress" ip="127.0.0.*" matchType="equals"/>
</authFilter>
```

### Excluding patterns

The following example shows how to use a piped list of values for the `requestUrl` element. Matching any of the patterns in the list is sufficient to satisfy the requirements of that particular element. In this example, the request URL must contain either `/SimpleServlet`, `/EmployeeRoleServlet`, or `/AllRoleServlet`. In addition, the request URL must not contain `/ManagerRoleServlet` and the request must come from an Internet Explorer user agent.

```
<authFilter id="myAuthFilter">
 <requestUrl id="myURL1" urlPattern="/SimpleServlet|/EmployeeRoleServlet|/AllRoleServlet" matchType="contains"/>
 <requestUrl id="myURL2" urlPattern="/ManagerRoleServlet" matchType="notContain" />
 <userAgent id="myAgent" agent="IE" matchType="contains" />
</authFilter>
```

### Example using all sub-elements

To process an incoming request with the service configured to use this filter, the request must meet the following conditions:

- Contains the pattern `/SimpleServlet` in the request URL
- Targets a domain that contains `host.example.com`
- Comes from the IP address `127.0.0.1`
- Comes from a Firefox browser
- The name of the target application is `myApp`

```
<authFilter id="myAuthFilter">
 <requestUrl id="myRequestUrl" urlPattern="/SimpleServlet" matchType="contains"/>
 <host id="myHost" name="host.example.com" matchType="contains"/>
 <remoteAddress id="myAddress" ip="127.0.0.1" matchType="equals"/>
 <userAgent id="myUserAgent" agent="Firefox" matchType="equals"/>
 <webApp id="myWebApp" name="myApp" matchType="contains"/>
</authFilter>
```

---

## Authorizing access to resources in Liberty

The purpose of authorization is to determine whether a user or group has the necessary privileges to access a resource.

### About this task

To learn about how authorization works in Liberty, see “Authorization” on page 606.

The following topics are covered in this section:

### Procedure

Configure authorization for applications in a Liberty server

## Configuring authorization for applications in Liberty

Configuring authorization for your application is to verify whether a user or group belongs to a specified role, and whether this role has the privilege to access a resource.

### About this task

The Liberty server extracts user and group mapping information from a user registry, then checks the authorization configuration for the application to determine whether a user or group is assigned to one of the required roles. Then the server reads the deployment descriptor of the application, to determine whether the user or group has the privilege to access the resource.

### Procedure

1. Enable the appSecurity-2.0 Liberty feature in the server.xml file.

For example:

```
<featureManager>
 <feature>appSecurity-2.0</feature>
</featureManager>
```

2. Configure a user registry for authentication on the Liberty server.  
See “Authenticating users in Liberty” on page 1168.
3. Ensure that the deployment descriptor for your application includes security constraints and other security related information.

**Note:** You can also use a tool such as Rational Application Developer to create the deployment descriptor.

4. Configure the authorization information such as the user and group to role mapping.

You can configure the authorization table in the following ways:

- If you have an EAR file, you can add the authorization configuration definition to the `ibm-application-bnd.xml` or `ibm-application-bnd.xmi` file.
- If you have standalone WAR files, you can add the authorization table definitions to the `server.xml` file under the respective application element. You can use the WebSphere Application Server Developer Tools for Eclipse to do this.

#### Notes:

- If you have an EAR file, the authorization configuration might already exist. In EAR files that are written to the current specification, this information is stored in an `ibm-application-bnd.xml` file; in older EAR files, this information is stored in an `ibm-application-bnd.xmi` file.

- If your EAR file does not already contain an `ibm-application-bnd.xml` file, it is not a straightforward task to create one and you might prefer to add the authorization configuration to the `server.xml` file.
- If the authorization configuration for the EAR file is defined in an `ibm-application-bnd.xml` file and also in the `server.xml` file, then the two tables are merged. If there are any conflicts, the information from the `server.xml` file is used.
- If you modify your user registry, be sure to review the authorization table for necessary changes. For example, if you are specifying an `access-id` element and change the realm name of the registry, you must also change the realm name in the `access-id` element.
- If you specify the `application-bnd` element in the `server.xml` file, your application must not be in the `dropins` folder. If you leave it in the `dropins` folder, then you must disable application monitoring by setting the following in your `server.xml` file:

```
<applicationMonitor dropinsEnabled="false" />
```

A role can be mapped to a user, a group, or a special subject. The two types of special subject are `EVERYONE` and `ALL_AUTHENTICATED_USERS`. When a role is mapped to the `EVERYONE` special subject, there is no security because everyone is allowed access and you are not prompted to enter credentials. When a role is mapped to the `ALL_AUTHENTICATED_USERS` special subject, then any user who has been authenticated by the application server can access the protected resource.

Here is example code for configuring the user and group to role mapping in the `server.xml` file:

```
<application type="war" id="myapp" name="myapp" location="${server.config.dir}/apps/myapp.war">
 <application-bnd>
 <security-role name="user">
 <group name="students" />
 </security-role>
 <security-role name="admin">
 <user name="gjones" />
 <group name="administrators" />
 </security-role>
 <security-role name="AllAuthenticated">
 <special-subject type="ALL_AUTHENTICATED_USERS" />
 </security-role>
 </application-bnd>
</application>
```

In this example, the `admin` role is mapped to the user ID `gjones` and all users in the group `administrators`. The `AllAuthenticatedRole` is mapped to the special subject `ALL_AUTHENTICATED_USERS`, meaning that any user has access as long as they provide valid credentials for authentication.

## Configuring security authorization for Liberty servers on IBM i

IBM i

Using the `iAdmin GRANTAUTH` command, you can authorize the `QEJBSVR` user profile to access the required resources for running the Liberty server.

### Before you begin

Servers run under the `QEJBSVR` user profile if one of the following is true:

- The Liberty environment was installed as a feature of a product offering using the IBM Installation Manager.
- the Job Manager was used to install the Liberty environment and the **Run optional installation scripts on IBM i targets** option is selected. See [Installing Liberty resources using the job manager](#).
- The `iAdmin POSTINSTALL` command was called after Installing Liberty by extracting an archive file.

Also, `QEJBSVR` is granted authorization to files in the `$WLP_USER_DIR` and `$WLP_OUTPUT_DIR` locations in all of these installation scenarios. Additionally, when servers are created, `QEJBSVR` is granted authorization

to server definition files and the \$WLP\_OUTPUT\_DIR location.

## About this task

This task provides example commands that show you how to authorize the QEJBSVR user profile to access the required resources for running the server after doing the following tasks:

- Creating files manually or modifying the authorities on shared resources and server definitions files.
- Configuring a server to access resources the QEJBSVR user profile is not yet authorized to.

## Example

- Granting the server role to the QEJBSVR user profile for the shared resources, server definitions and output locations configured for the Liberty environment installed at /WAS/wlp directory.  

```
/WAS/wlp/lib/native/os400/bin/iAdmin GRANTAUTH -rolename server -userprofile QEJBSVR
```
- Granting the server role to the QEJBSVR user profile for shared resources and server definitions in /WAS/myWlpServers/usr, and for any server output locations defined by the **WLP\_OUTPUT\_DIR** variable in files matching the definition in the /WAS/myWlpServers/usr/servers/\*/server.env file.  

```
/WAS/wlp/lib/native/os400/bin/iAdmin GRANTAUTH -rolename server -userprofile QEJBSVR
-userdir /WAS/myWlpServers/usr
```
- Granting the server role to the QEJBSVR user profile for output location /WAS/myWlpOutput/servers.  

```
/WAS/wlp/lib/native/os400/bin/iAdmin GRANTAUTH -rolename server -userprofile QEJBSVR
-outputdir /WAS/myWlpOutput/servers
```

## OAuth

OAuth is an open standard for delegated authorization. With the OAuth authorization framework, a user can grant a third-party application access to their information stored with another HTTP service without sharing their access permissions or the full extent of their data.

In OAuth, the client, or third-party application, requests access to resources controlled by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource owner. Instead of using the credentials of the resource owner to access protected resources, the client obtains an access token, which is a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server.

OAuth 2.0 is not compatible with OAuth 1.0. OAuth 2.0 provides ease of use for client application developers, and authorization flows for different types of client applications.

WebSphere Application Server supports OAuth 2.0, and can be used as an OAuth service provider endpoint and an OAuth protected resource enforcement endpoint.

WebSphere Application Server supports the following OAuth standard specifications:

- The OAuth 2.0 Authorization Framework
- The OAuth 2.0 Authorization Framework: Bearer Token Usage

The following list shows a summary of features within WebSphere Application Server OAuth 2.0 services.

- WebSphere Application Server acts as an OAuth Service Provider (SP) to handle OAuth 2.0 protocol requests.
- WebSphere Application Server acts as protected resource enforcement endpoint to authorize or deny requests for deployed web resources.
- Allow multiple service providers to co-exist.
- Allow administrator to revoke access tokens.



- Allow client to revoke its authorization given by a user.
- Optionally provide a Subject for a resource application to make an authenticated downstream call or perform programmatic J2EE security.
- Support 4 typical OAuth 2.0 flows as defined in the protocol.
- Support persistent OAuth services.

## OAuth 2.0 services

WebSphere Application Server OAuth services include both the OAuth authorization service and the web resource authorization decision service.

The OAuth 2.0 authorization service provides all OAuth 2.0 protocol endpoint URLs, and is responsible for client authorization and token issuing.

The web resource authorization decision service is built into the Liberty web authentication code. When a client accesses an OAuth protected web resource, the OAuth token is validated and mapped to a WebSphere Application Server platform security subject that the web request then runs under.

### Defining an OAuth service provider:

An OAuth service provider is a named set of configuration options for OAuth. The id or name of the provider is specified in the URL of inbound requests to the authorization and token endpoints. The set of configuration options for that provider is used when the request is handled. This process allows one server with one endpoint servlet to effectively provide multiple OAuth configurations. For example, the `https://my.company.com:8021/oauth2/endpoint/photoShare/authorize` URL is handled by using the set of OAuth configuration options that are defined for the OAuth provider named `photoShare`. The `https://my.company.com:8021/oauth2/endpoint/calendarAuthz/authorize` URL is handled by using the set of OAuth configuration options that are defined for the OAuth provider named `calendarAuthz`.

### About this task

An OAuth service provider is defined with the `oauthProvider` element in the `server.xml` file. You can define an OAuth service provider by editing the `server.xml` file or by using the WebSphere Application Server Development Tools for Liberty. This task describes how to define a minimal OAuth configuration.

### Procedure

1. Add the `oauth-2.0` and `ssl-1.0` features. OAuth is a secure protocol so SSL is required. On Liberty, you must supply a keystore password for SSL by using the `keyStore` element. There is no default keystore password.

```
<featureManager>
 <feature>oauth-2.0</feature>
 <feature>ssl-1.0</feature>
</featureManager>
```

2. Set up the role mapping for the OAuth web application by using the `oauth-roles` element. OAuth is an HTTP-based protocol and a web application is supplied to handle the authorization and token endpoints. The web application is built in and is started automatically when you specify the `oauth-2.0` feature. However, you must map the authenticated role to one or more users, groups, or special subjects. Another role, `clientManager`, is supplied for managing client configuration, but it is not necessary to map that role for OAuth authorization to function.

```
<oauth-roles>
 <authenticated>
 <user>testuser</user>
 </authenticated>
</oauth-roles>
```

3. Define one or more providers with the `oauthProvider` element. The provider must have at least one client defined. Clients can be defined locally with the `localStore` and `client` elements. Clients can also be defined in a relational database with the `databaseStore` element.

```
<oauthProvider id="SampleProvider" filter="request-url%=ssodemo">
 <localStore>
 <client name="client01" secret="{xor}LDo8LTor"
 displayname="Test client number 1"
 redirect="http://localhost:1234/oauthclient/redirect.jsp"
 enabled="true" />
 </localStore>
</oauthProvider>
```

4. Define a user registry, either an LDAP registry by specifying the `ldapRegistry-3.0` feature and the `ldapRegistry` configuration element, or a basic registry by specifying the `basicRegistry` configuration element.

```
<basicRegistry id="basic" realm="BasicRealm">
 <user name="testuser" password="testuserpwd" />
</basicRegistry>
```

5. Set the `allowFailOverToBasicAuth` web application security property to true.

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

## Results

You have defined a minimal OAuth configuration.

## Example

The following example shows a sample `server.xml` file that defines a simple OAuth provider with one client:

```
<server>

 <featureManager>
 <feature>oauth-2.0</feature>
 <feature>ssl-1.0</feature>
 </featureManager>

 <keyStore password="keypass" />

 <oauth-roles>
 <authenticated>
 <user>testuser</user>
 </authenticated>
 </oauth-roles>

 <oauthProvider id="SampleProvider" filter="request-url%=ssodemo">
 <localStore>
 <client name="client01" secret="{xor}LDo8LTor"
 displayname="Test client number 1"
 redirect="http://localhost:1234/oauthclient/redirect.jsp"
 enabled="true" />
 </localStore>
 </oauthProvider>

 <webAppSecurity allowFailOverToBasicAuth="true" />

 <basicRegistry id="basic" realm="BasicRealm">
 <user name="testuser" password="testuserpwd" />
 </basicRegistry>

</server>
```

## OAuth full profile provider configuration equivalents:

The following tables map the Liberty server.xml file elements and attributes to the equivalent full profile provider parameters in the provider configuration file.

The following table illustrates the Liberty profile attributes and the equivalent full profile parameters for the oauthProvider element.

Table 90. Liberty oauthProvider element

Liberty attribute name	Full profile equivalent parameter
authorizationGrantLifetime	oauth20.max.authorization.grant.lifetime.seconds
authorizationCodeLifetime	oauth20.code.lifetime.seconds
authorizationCodeLength	oauth20.code.length
accessTokenLifetime	oauth20.token.lifetime.seconds
accessTokenLength	oauth20.access.token.length
issueRefreshToken	oauth20.issue.refresh.token
refreshTokenLength	oauth20.refresh.token.length
mediatorClassname	oauth20.mediator.classnames
allowPublicClients	oauth20.allow.public.clients
grantType	oauth20.grant.types.allowed
authorizationFormTemplate	oauth20.authorization.form.template
authorizationErrorTemplate	oauth20.authorization.error.template
customLoginURL	oauth20.authorization.loginURL
autoAuthorizeParam	oauth20.autoauthorize.param
autoAuthorizeClient	oauth20.autoauthorize.clients
clientURISubstitutions	oauth20.client.uri.substitutions
filter	Filter
oauthOnly	oauthOnly
includeTokenInSubject	includeToken
characterEncoding	characterEncoding
clientTokenCacheSize	oauth20.token.userClientTokenLimit

The following table illustrates the Liberty attributes and the equivalent full profile parameters for the databaseStore element.

Table 91. Liberty databaseStore element

Liberty attribute name	Full profile equivalent parameter
cleanupExpiredTokenInterval	oauthjdbc.CleanupInterval

## Configuring automatic authorization: Before you begin

Ensure that you enabled the OAuth 2.0 feature and configured an OAuth service provider by following Defining an OAuth service provider.

### About this task

You can authorize an OAuth client without the approval of the resource owner by enabling the automatic authorization feature of the WebSphere Application Server OAuth service provider.

## Procedure

To configure auto consent, use the `autoAuthorizeParam` attribute and the `<autoAuthorizeClient>` subelement of the `<oauthProvider>` element in the `server.xml` file:

```
<oauthProvider id="OAuthConfigSample" autoAuthorizeParam="autoauthz" ...>
 ...
 <autoAuthorizeClient>client01</autoAuthorizeClient>
 <autoAuthorizeClient>client02</autoAuthorizeClient>
</oauthProvider>
```

## Results

The `client01` and `client02` OAuth clients are configured for automatic authorization.

### OAuth endpoint URLs:

After OAuth 2.0 is enabled, several endpoint URLs are configured on your WebSphere Application Server so that OAuth clients can communicate with the OAuth service provider before accessing OAuth protected resources.

The following endpoint URLs are configured for the OAuth service provider:

- Authorization endpoint URL

`https://host_name:port_number/oauth2/endpoint/provider_name/authorize`

where

- `host_name` is the host name of the OAuth service provider.
- `port_number` is the secure port number that is configured on the WebSphere Application Server.
- `provider_name` is the OAuth provider name.

- Token endpoint URL

`https://host_name:port_number/oauth2/endpoint/provider_name/token`

where

- `host_name` is the host name of the OAuth service provider.
- `port_number` is the secure port number that is configured on the WebSphere Application Server.
- `provider_name` is the OAuth provider name.

- Authorization endpoint URL of trust association interceptor (TAI) based user authentication

`https://host_name:port_number/oauth2/declaritiveEndpoint/provider_name/authorize`

where

- `host_name` is the host name of the OAuth service provider.
- `port_number` is the secure port number that is configured on the WebSphere Application Server.
- `provider_name` is the OAuth provider name.

By using this authorization endpoint, the applicable user authentication includes TAI.

### OAuth 2.0 service invocation

A registered OAuth client can invoke the WebSphere Application Server OAuth service authorization endpoint to request an access token. A registered OAuth client can also invoke the WebSphere Application Server OAuth service token endpoint to request an access token. The client then can use the access token to request protected web resources from WebSphere Application Server.

WebSphere Application Server OAuth 2.0 service supports the following flows.

## Authorization code flow

Invoke authorization endpoint to request authorization code.

The OAuth client redirects the resource owner or user to the WebSphere Application Server OAuth 2.0 Authorization Service by adding its client id, client secret, state, redirect URI, and the optional scopes.

```
https://host_name:port_number/oauth2/endpoint/provider_name/authorize
```

or

```
https://host_name:port_number/oauth2/declarativeEndpoint/provider_name/authorize
```

Invoke OAuth token endpoint to request access token.

The OAuth client requests an access token from the WebSphere Application Server OAuth 2.0 token endpoint by adding authorization\_code grant type, authorization code, redirect\_url, and client\_id as request parameters.

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

The following example shows the constructions of the URIs when using authorization code, and the use of the access token to access web resources:

```
String charset = "UTF-8";
String param1 = "code";

if (isAuthorizationCode){
 String query = String.format("response_type=%s&
 client_id=%s&
 client_secret=%s&
 state=%s&
 redirect_uri=%s&
 scope=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(clientSecret, charset),
 URLEncoder.encode(state, charset),
 URLEncoder.encode(redirectURI, charset),
 URLEncoder.encode(scope, charset));

 String s = authorizationEndPoint + "?" + query;
 System.out.println("Visit: " + s + "\nand grant permission");
 System.out.print("Now enter the OAuth code you have received in redirect uri :");
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 String code = br.readLine();
 param1 = "authorization_code";
 query = String.format("grant_type=%s&
 code=%s&
 client_id=%s&
 client_secret=%s&
 state=%s&
 redirect_uri=%s&
 scope=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(code, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(clientSecret, charset),
 URLEncoder.encode(state, charset),
 URLEncoder.encode(redirectURI, charset),
 URLEncoder.encode(scope, charset));

 URL url = new URL(tokenEndPoint);
 HttpURLConnection con = (HttpURLConnection)url.openConnection();
 con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded;charset=" + charset);
 con.setDoOutput(true);
 con.setRequestMethod("POST");
 OutputStream output = null;
 try {
 output = con.getOutputStream();
 output.write(query.getBytes(charset));
 output.flush();
 } finally {
 if (output != null) try {
 output.close();
 }
 }
}
```

```

 } catch (IOException logOrIgnore) {}
}
con.connect();
System.out.println("response message is = " + con.getResponseMessage());
// read the output from the server
BufferedReader reader = null;
StringBuilder stringBuilder;
reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
stringBuilder = new StringBuilder();
String line = null;
try {
 while ((line = reader.readLine()) != null) {
 stringBuilder.append(line + "\n");
 }
} finally {
 if (reader != null) try {
 reader.close();
 } catch (IOException logOrIgnore) {}
}
String tokenResponse = stringBuilder.toString();
System.out.println ("response is = " + tokenResponse);
JSONObject json = JSONObject.parse(tokenResponse);
if (json.containsKey("access_token")) {
 accessToken = (String)json.get("access_token");
 this.accessToken = accessToken;
}
if (json.containsKey("refresh_token")) {
 refreshToken = (String)json.get("refresh_token");
}
//sendRequestForAccessToken(query);
if (accessToken != null) {
 String query = String.format("access_token=%s",
 URLEncoder.encode(accessToken, charset));
 URL urlResource = new URL(resourceEndPoint);
 HttpsURLConnection conn = (HttpsURLConnection) urlResource.openConnection();
 conn.setRequestMethod("POST");
 conn.setRequestProperty("Content-type", "application/x-www-form-urlencoded");
 conn.setDoOutput(true);
 output = null;
 try {
 output = conn.getOutputStream();
 output.write(query.getBytes(charset));
 output.flush();
 } finally {
 if (output != null) try {
 output.close();
 } catch (IOException logOrIgnore) {}
 }
}
conn.connect();
System.out.println("response to the resource request is = " + conn.getResponseMessage ());
reader = null;
if(conn.getResponseCode()>=200 && conn.getResponseCode() < 400) {
 reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
 stringBuilder = new StringBuilder();
 String line = null;
 try {
 while ((line = reader.readLine()) != null) {
 stringBuilder.append(line + "\n");
 }
 } finally {
 if (reader != null) try {
 reader.close();
 } catch (IOException logOrIgnore) {}
 }
}
System.out.println ("response message to the request resource is = " + stringBuilder.toString());
} else {
 isValidResponse = false;
}
}
}
}

```

## Implicit grant flow

The OAuth client requests an access token from the WebSphere Application Server OAuth 2.0 authorization endpoint by adding token\_response\_type, redirect\_url, client\_id, scope, and state as request parameters.

```
https://host_name:port_number/oauth2/endpoint/provider_name/authorize
```

or

```
https://host_name:port_number/oauth2/declarativeEndpoint/provider_name/authorize
```

The following example shows the construction of the URI when using implicit grant:

```
if (isImplicit) {
 param1 = "token";
 String query = String.format("response_type=%s&
 client_id=%s&
 state=%s&
 redirect_uri=%s&
 scope=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(state, charset),
 URLEncoder.encode(redirectURI, charset),
 URLEncoder.encode(scope, charset));

 String s = authorizationEndPoint + "?" + query;
 System.out.println("Visit: " + s + "\nand grant permission");
 System.out.print("Now enter the access token you have received in redirect uri :");
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 accessToken = br.readLine();
 if (accessToken != null) {
 // send Resource Request using the access token
 }
}
```

### Client credential flow

The OAuth client accesses the token endpoint by using the client ID and client secret, and exchanges for an access token for future resource requests. In this flow, the client accesses the token endpoint by adding `client_credentials` grant type, `client_id`, and `client_secret` as request parameters.

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

The following example shows the construction of the URI when using client credential:

```
if (isClientCredentials){
 param1 = "client_credentials";
 String query = String.format("grant_type=%s&
 scope=%s&
 client_id=%s&
 client_secret=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(scope, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(clientSecret, charset));

 accessToken = sendRequestForAccessToken(query);
 if (accessToken != null) {
 //send Resource Request using (accessToken);
 }
}
```

### Resource owner password credentials flow

The Resource Owner Password Credentials flow passes the user ID and password of the resource owner to the token endpoint directly. In this flow, The OAuth client accesses the token endpoint by adding `password` grant type, `client_id`, `client_secret`, `username`, `password`, `scope`, and `state` as request parameters.

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

The following example shows the construction of the URI when using resource owner password:

```
if (isResourceOwnerCredentials) {
 param1 = "password";
 String query = String.format("grant_type=%s&
 username=%s&
 password=%s&
 scope=%s&
 client_id=%s&
 client_secret=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(resOwnerName, charset),
 URLEncoder.encode(resOwnerPassword, charset),
 URLEncoder.encode(scope, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(clientSecret, charset));
 accessToken = sendRequestForAccessToken(query);
 if (accessToken != null) {
 //send Resource Request using (accessToken);
 }
}
```

If the access token is expired, then the refresh token can be sent to get a valid access token. The following example shows how to send a refresh token:

```
if(isAccessToken) {
 if (this.accessToken != null) {
 if (!sendResourceRequest(this.accessToken)) {
 // resource request failed...
 //get refresh token
 param1 = "refresh_token";
 String query = String.format("grant_type=%s&
 client_id=%s&
 client_secret=%s&
 refresh_token=%s&
 scope=%s",
 URLEncoder.encode(param1, charset),
 URLEncoder.encode(clientId, charset),
 URLEncoder.encode(clientSecret, charset),
 URLEncoder.encode(this.refreshToken, charset),
 URLEncoder.encode(scope, charset));
 accessToken = sendRequestForAccessToken(query);
 if (accessToken != null) {
 sendResourceRequest(accessToken);
 }
 }
 }
}
```

## Customizing an OAuth provider

The WebSphere Application Server OAuth service provider has plug-in points for customization. You can replace the default form login page for user authentication, or develop your own user consent form to collect client authorization data. WebSphere Application Server OAuth providers also allow customized post processing for major events in OAuth token issuing by using mediators.

### Custom mediator:

An OAuth 2.0 mediator is used as a callback during the OAuth 2.0 message processing to perform customized post processing.



## Write an OAuth 2.0 mediator

To write a mediator, you must implement the interface named `com.ibm.oauth.core.api.oauth20.mediator.OAuth20Mediator`. You can implement one or more of the following methods to perform custom post processing.

```
void init(OAuthComponentConfiguration config)
```

This method is called by a factory when an instance of this object is created.

```
void mediateAuthorize(AttributeList attributeList)
```

This method is called by the core component after basic message validation and processing to allow any post custom processing by the component consumer in the `processAuthorization` method.

```
void mediateAuthorizeException(AttributeList attributeList, OAuthException exception)
```

This method is called by the core component when the protocol exception happens to allow any post custom processing by the component consumer in the `processAuthorization` method.

```
void mediateResource(AttributeList attributeList)
```

This method is called by the core component after basic message validation and processing to allow any post custom processing by the component consumer in the `processResourceRequest` method.

```
void mediateResourceException(AttributeList attributeList, OAuthException exception)
```

This method is called by the core component when protocol exception happens to allow any post custom processing by the component consumer in the `processResourceRequest` method.

```
void mediateToken(AttributeList attributeList)
```

This method is called by the core component after basic message validation and processing to allow any post custom processing by the component consumer in the `processTokenRequest` method.

```
void mediateTokenException(AttributeList attributeList, OAuthException exception)
```

This method is called by the core component when protocol exception happens to allow any post custom processing by the component consumer in the `processTokenRequest` method.

## Enable OAuth 2.0 mediator for an OAuth provider

To add a customized mediator to a specific OAuth 2.0 service provider, update the provider definition in the `server.xml` file. Add the `mediatorClassname` attribute of the `oauthProvider` element and specify the class name for the mediator. You can also specify multiple class names for mediators by using the `mediatorClassname` subelement of the `oauthProvider` element. If multiple mediators are specified, those mediators are started in the order they are specified. You must also define a library element that contains the mediator class and refer to the library with the `libraryRef` attribute.

The following example shows a sample custom mediator entry in the provider definition in the `server.xml` file:

```
<oauthProvider id="OAuthConfigSample" libraryRef="myLib"
 mediatorClassname="com.ibm.ws.security.oauth20.mediator.ResourceOwnerValidationMediator" ...>
 ...
</oauthProvider>

<library id="myLib">
 <fileset dir="C:\mydir" includes="myLib.jar" />
</library>
```

The following code sample implements the credential validation by using the WebSphere Application Server user registry in the resource owner password credentials flow.

```

package com.ibm.ws.security.oauth20.mediator;

import com.ibm.oauth.core.api.attributes.AttributeList;
import com.ibm.oauth.core.api.config.OAuthComponentConfiguration;
import com.ibm.oauth.core.api.error.OAuthException;
import com.ibm.oauth.core.api.error.oauth20.OAuth20MediatorException;
import com.ibm.oauth.core.api.oauth20.mediator.OAuth20Mediator;
import com.ibm.oauth.core.internal.oauth20.OAuth20Constants;
import com.ibm.websphere.security.CustomRegistryException;
import com.ibm.websphere.security.PasswordCheckFailedException;
import com.ibm.websphere.security.UserRegistry;

import java.rmi.RemoteException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ResourceOwnerValidationMediator implements OAuth20Mediator {
 private static final String CLASS = ResourceOwnerValidationMediator.class.getName();
 private static final Logger LOG = Logger.getLogger(CLASS);
 private UserRegistry reg = null;

 public void init(OAuthComponentConfiguration config) {
 try {
 InitialContext ctx = new InitialContext();
 reg = (UserRegistry) ctx.lookup("UserRegistry");
 } catch (NamingException ne) {
 LOG.log(Level.SEVERE, "Cannot lookup UserRegistry", ne);
 }
 }

 public void mediateAuthorize(AttributeList attributeList)
 throws OAuth20MediatorException {
 // nothing to do here
 }

 public void mediateAuthorizeException(AttributeList attributeList,
 OAuthException exception)
 throws OAuth20MediatorException {
 // nothing to do here
 }

 public void mediateResource(AttributeList attributeList)
 throws OAuth20MediatorException {
 // nothing to do here
 }

 public void mediateResourceException(AttributeList attributeList,
 OAuthException exception)
 throws OAuth20MediatorException {
 // nothing to do here
 }

 public void mediateToken(AttributeList attributeList)
 throws OAuth20MediatorException {
 final String methodName = "mediateToken";
 LOG.entering(CLASS, methodName, attributeList);
 if("password".equals(attributeList.getAttributeValueByName("grant_type"))) {
 String username = attributeList.getAttributeValueByName("username");
 String password = attributeList.getAttributeValueByName("password");
 try {

```

```

 reg.checkPassword(username, password);
 } catch (PasswordCheckFailedException e) {
 throw new OAuth20MediatorException("User doesn't exist or the
 password doesn't match.", e);
 } catch (CustomRegistryException e) {
 throw new OAuth20MediatorException("Cannot validate resource owner.", e);
 } catch (RemoteException e) {
 throw new OAuth20MediatorException("Cannot validate resource owner.", e);
 }
}
LOG.exiting(CLASS, methodName);
}

public void mediateTokenException(AttributeList attributeList,
 OAuthException exception)
 throws OAuth20MediatorException {
 final String methodName = "mediateTokenException";
 LOG.entering(CLASS, methodName, new Object[] {attributeList, exception});
 if("password".equals(attributeList.getAttributeValueByName("grant_type"))) {
 // clear sensitive data
 attributeList.setAttribute("access_token",
 OAuth20Constants.ATTRTYPE_RESPONSE_ATTRIBUTE,
 new String[0]);
 attributeList.setAttribute("refresh_token",
 OAuth20Constants.ATTRTYPE_RESPONSE_ATTRIBUTE,
 new String[0]);
 }
 LOG.exiting(CLASS, methodName);
}
}
}

```

### Custom consent form template:

The OAuth authorization server provides a template to acquire user consent information about which OAuth clients are authorized to access the protected resource in given scopes. The authorization request from the OAuth client includes a list of requested scopes from the template.

WebSphere Application Server allows the consent form template to be either a static HTML page or a dynamic web page. In both cases, the template must be provided as an unprotected web resource. The form retriever in WebSphere Application Server integration does not perform any authentication when accessing this template URL.

The WebSphere Application Server OAuth provider includes a sample consent form template, and allows customization by using `oauthFormData` variable.

To customize the consent form, you must edit the `oauthFormData` variable by using JavaScript. The following variables are included in the form data:

- `authorizationUrl` - the authorization URL where the form is being submitted
- `clientDisplayName` - the display name of the client
- `nonce` - random generated number to prevent cross-site request forgery (CSRF)
- `client_id` - see the OAuth 2.0 specification
- `response_type` - see the OAuth 2.0 specification
- `redirect_uri` - see the OAuth 2.0 specification
- `state` - see the OAuth 2.0 specification
- `scope` - see the OAuth 2.0 specification

The developer of a form template must include the content of the `oauthFormData` variable, by using JavaScript. The developer must interpret the scope value to be a meaningful value to a user. When a user authorizes the request, the developer can call the `submitForm(oauthFormData)` method to perform the authorization. The `submitForm` method is provided by default. However, if developers are familiar with OAuth 2.0 protocol, they can implement their own function to submit the OAuth authorization request.



8.5.5.4

A `cancel(oauthFormData)` method is provided by default and can be used to allow a user to cancel the authorization request.



8.5.5.4

The consent form can also be modified to allow the user's consent selection to be cached. This means that if the same OpenID Connect client makes a new authorization request with the same approved scopes or reduced scopes, the user is not be prompted with the consent form. Instead, the previously allowed scopes are considered authorized and passed to the protected resource accordingly.



8.5.5.4

If client registration is in `localStorage` mode, the user's consent selection is cached in the browser session. The approved scopes remain cached for the given user until the session is closed or until a set amount of time (specified in the server configuration) has elapsed.



8.5.5.4

If client registration is in `databaseStore` mode, the user's consent selection can be persistent in a database table, `OAuthDBSchema.OAUTH20CONSENTCACHE`. The scopes remain cached for the given user until a set amount of time (specified in the server configuration) has elapsed. The OpenID Connect provider tries to create the consent cache table automatically, but it is suggested that the user explicitly create the consent table when configuring a database for an OAuth2.0 provider and OpenID Connect provider, see [Persistent OAuth service configuration](#) for further details.



8.5.5.4

To use this functionality, the developer of a form template must include a `prompt` value within the `oauthFormData` JavaScript object. In order to cache the user's affirmative response and prevent the consent form from being displayed again in the same session, the `prompt` value is set to the string `none`. To allow the user to submit an affirmative response without caching the approval, the `prompt` value is set to the string `consent`.

You can use a dynamic page that returns globalized content according to the `Accept-Language` header in the request. When retrieving the template, the `Accept-Language` header is forwarded, and the template developer must decide which content to return regarding the preferred language.

**Note:** The `clientDisplayName` variable is not escaped in HTML. The template developer must sanitize the value, as the value is input by a user during client registration.

To use a custom consent form template page for a specific OAuth 2.0 service provider, you must update the service provider definition in the `server.xml` file. In the provider configuration, you must use the `authorizationFormTemplate` attribute of the `oauthProvider` element and add the template URL as the value. The following example shows a sample template entry in the provider configuration:

```
<oauthProvider id="OAuthConfigSample"
 authorizationFormTemplate="https://acme.com:9043/oath20/template.html
 ...>
```

The following example illustrates a sample consent form:

```
<oauthProvider id="OAuthConfigSample"
 authorizationLoginURL="https://acme.com:9043/oath20/login.jsp"
 ...>
```

```
function escapeHTML(str) {
 var ele = document.createElement("div");
 ele.innerText = ele.textContent = str;
 return ele.innerHTML;
}
```

```

}
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>OAuth authorization form</title>
<script language="javascript">
function init() {
var scope = oauthFormData.scope;
var scopeEle = document.getElementById("oauth_scope");
var ul = document.createElement("ul");
if(scope) {
for(var i=0; i< scope.length; i++) {
var n = document.createElement("li");
n.innerHTML = scope[i];
ul.appendChild(n);
}
}
scopeEle.appendChild(ul);
// set client name
var clientEle = document.getElementById("client_name");
clientEle.innerHTML = escapeHTML(oauthFormData.clientDisplayName);
}

function escapeHTML(str) {
var ele = document.createElement("div");
ele.innerText = ele.textContent = str;
return ele.innerHTML;
}
</script>
</head>
<body onload="init()">
<div>Do you want to allow client xxxxxxx to access your data?</div>
<div id="oauth_scope">
</div>
<div>
<form action="javascript:submitForm(oauthFormData);">
<input type="submit" value="Allow, remember my decision" onclick="javascript:oauthFormData.prompt = 'none';"/>
<input type="submit" value="Allow once" onclick="javascript:oauthFormData.prompt = 'consent';"/>
<input type="button" value="Cancel" onclick="javascript:cancel(oauthFormData);"/>
</form>
</div>
</body>
</html>

```

### Custom user login form:

The WebSphere Application Server OAuth service provider includes a form login page for a user to submit a user name and password.

You can customize your own form login page, but it must be implemented as required in the form-based authentication in the servlet specification. In this form, the action must be `j_security_check`, and use the `j_username` input field to get the user name. The action must also use the `j_password` input field to get the user password. The custom form login page must be provided as an unprotected web resource.

To use the custom form login page for a specific OAuth20 service provider, you must update the service provider definition in the `server.xml` file. In the provider configuration, you must add the `customLoginURL` attribute and specify the login page URL as the value.

The following is an example custom login page entry in the provider definition:

```

<oauthProvider id="OAuthConfigSample"
 customLoginURL="https://acme.com:9043/oauth20/login.jsp"
 ...>

```

**Note:** Make sure that any files included in your form-login page (such as external style sheets, or images) are unprotected.

## Persistent OAuth service configuration

WebSphere Application Server supports a persistent OAuth 2.0 service by persisting OAuth tokens and clients in a database. With persistent OAuth 2.0 services, an authorized client can access OAuth 2.0 service after OAuth services are restarted.

To configure persistent OAuth 2.0 services, complete the following steps:

### 1. Configure the OAuth 2.0 service provider.

To use a database store, you must specify the <databaseStore> subelement of the <oauthProvider> element. The only required attribute on the <databaseStore> element is <dataSourceRef>, whose value must be the id of the <dataSource> element.

The following example is a sample server.xml file for an OAuth provider that uses a Derby database store:

```
<server>

 <featureManager>
 <feature>oauth-2.0</feature>
 <feature>ssl-1.0</feature>
 <feature>jdbc-4.0</feature>
 <feature>jndi-1.0</feature>
 </featureManager>

 <keyStore password="keyspass" />

 <oauth-roles>
 <authenticated>
 <user>testuser</user>
 </authenticated>
 </oauth-roles>

 <oauthProvider id="OAuthConfigDerby" filter="request-url%=ssodemo"
 oauthOnly="false">
 <databaseStore dataSourceRef="OAuthFvtDataSource" />
 </oauthProvider>

 <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib" />

 <library id="DerbyLib" fileSetRef="DerbyFileset" />

 <fileset id="DerbyFileset" dir="{DERBY_JDBC_DRIVER_PATH}"
 includes="derby.jar" />

 <dataSource id="OAuthFvtDataSource" jndiName="jdbc/OAuth2DB"
 jdbcDriverRef="DerbyEmbedded">
 <properties.derby.embedded databaseName="D:\oauth2db"
 createDatabase="create" />
 </dataSource>

 <webAppSecurity allowFailOverToBasicAuth="true" />

 <basicRegistry id="basic" realm="BasicRealm">
 <user name="testuser" password="testuserpwd" />
 </basicRegistry>

</server>
```

### 2. Set up a database and table to store the OAuth token and client.

- a. Create a database for persistent OAuth service. See the vendor documentation for database creation. In this example, the database name is D:\oauth2db.
- b. Create 3 OAuth tables as defined by the following SQL statements:

```

----- CREATE TABLES -----
CREATE TABLE OAuthDBSchema.OAUTH20CACHE
(
 LOOKUPKEY VARCHAR(256) NOT NULL,
 UNIQUEID VARCHAR(128) NOT NULL,
 COMPONENTID VARCHAR(256) NOT NULL,
 TYPE VARCHAR(64) NOT NULL,
 SUBTYPE VARCHAR(64),
 CREATEDAT BIGINT,
 LIFETIME INT,
 EXPIRES BIGINT,
 TOKENSTRING VARCHAR(2048) NOT NULL,
 CLIENTID VARCHAR(64) NOT NULL,
 USERNAME VARCHAR(64) NOT NULL,
 SCOPE VARCHAR(512) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 STATEID VARCHAR(64) NOT NULL,
 EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{} '
);

CREATE TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
(
 COMPONENTID VARCHAR(256) NOT NULL,
 CLIENTID VARCHAR(256) NOT NULL,
 CLIENTSECRET VARCHAR(256),
 DISPLAYNAME VARCHAR(256) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 ENABLED INT,
 CLIENTMETADATA CLOB NOT NULL DEFAULT '{} '
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE
(
 CLIENTID VARCHAR(256) NOT NULL,
 USERID VARCHAR(256),
 PROVIDERID VARCHAR(256) NOT NULL,
 SCOPE VARCHAR(1024) NOT NULL,
 EXPIRES BIGINT,
 EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{} '
);

----- ADD CONSTRAINTS -----
ALTER TABLE OAuthDBSchema.OAUTH20CACHE
 ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
 ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

----- CREATE INDEXES -----
CREATE INDEX OAUTH20CACHE_EXPIRES ON OAuthDBSchema.OAUTH20CACHE (EXPIRES ASC);

```

### 3. Configure WebSphere Application Server.

Configure the WebSphere Application Server data source. You must set the data source Java Naming and Directory Interface (JNDI) name to be jdbc/OAuth2DB. The JNDI name must match the jndiName attribute of the <dataSource> element in the server.xml file. Enter the database name, for example, D:\oauth2db.

For more information about the configuration of DB2 and Derby for OAuth persistent services, see IBM DB2 for persistent OAuth services and Derby database for persistent OAuth services. You can use them as a sample template to configure other databases.

### 4. Add the registered OAuth clients to the database.

To persist a client in a database, you must save the client to the database. The following SQL statements add the dbc1ient01 and dbc1ient02 OAuth clients to a Derby database:

```

CONNECT 'jdbc:derby:D:\oauth2db';
INSERT INTO OAuthDBSchema.OAUTH2CLIENTCONFIG VALUES
(
 'OAuthConfigDerby',
 'dbclient01',
 'secret',
 'dbclient01',
 'http://localhost:9080/oauthclient/redirect.jsp',
 1
),
(
 'OAuthConfigDerby',
 'dbclient02',
 'secret',
 'dbclient02',
 'http://localhost:9080/oauthclient/redirect.jsp',
 1
);
DISCONNECT CURRENT;

```

**Note:** **8.5.5.2** The Componentid must be the same as the id of the oauthProvider element in the server.xml file.

### Derby database for persistent OAuth services:

Derby database can be used for persistent OAuth services. For convenience and reference purposes, this topic documents the steps you need to configure Derby database, either remote or local to the OAuth service, for OAuth persistent service.

To configure Derby database for persistent OAuth services, complete the following steps:

#### 1. Create a database and tables.

Edit and run the following SQL statement to create an OAuth database and table:

```

--- Change oauth2db to the name you want for the database
--- Connect to Derby, choose one connection option to uncomment
--- if connecting to Derby as network server
--- CONNECT 'jdbc:derby://localhost:1527/oauth2db;create=true';

--- if connecting to embedded derby, you can change D:\oauth2db to location of database
--- CONNECT 'jdbc:derby:D:\oauth2db;create=true';

--- if creating tables in existing Derby database, remove the create=true parameter.

----- CREATE TABLES -----
CREATE TABLE OAuthDBSchema.OAUTH2OCACHE (
 LOOKUPKEY VARCHAR(256) NOT NULL,
 UNIQUEID VARCHAR(128) NOT NULL,
 COMPONENTID VARCHAR(256) NOT NULL,
 TYPE VARCHAR(64) NOT NULL,
 SUBTYPE VARCHAR(64),
 CREATEDAT BIGINT,
 LIFETIME INT,
 EXPIRES BIGINT,
 TOKENSTRING VARCHAR(2048) NOT NULL,
 CLIENTID VARCHAR(64) NOT NULL,
 USERNAME VARCHAR(64) NOT NULL,
 SCOPE VARCHAR(512) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 STATEID VARCHAR(64) NOT NULL,
 EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}')
);

CREATE TABLE OAuthDBSchema.OAUTH2CLIENTCONFIG (

```



```

 COMPONENTID VARCHAR(256) NOT NULL,
 CLIENTID VARCHAR(256) NOT NULL,
 CLIENTSECRET VARCHAR(256),
 DISPLAYNAME VARCHAR(256) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 ENABLED INT,
 CLIENTMETADATA CLOB NOT NULL DEFAULT '{} '
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE (
 CLIENTID VARCHAR(256) NOT NULL,
 USERID VARCHAR(256),
 PROVIDERID VARCHAR(256) NOT NULL,
 SCOPE VARCHAR(1024) NOT NULL,
 EXPIRES BIGINT,
 EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{} '
);

----- ADD CONSTRAINTS -----
ALTER TABLE OAuthDBSchema.OAUTH20CACHE
 ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
 ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

----- CREATE INDEXES -----
CREATE INDEX OAUTH20CACHE_EXPIRES ON OAUTHDBSCHEMA.OAUTH20CACHE (EXPIRES ASC);

DISCONNECT CURRENT;

```

Run the createTables.sql file by starting **ij** with the following command:

```
ij createTables.sql
```

## 2. Configure the WebSphere Application Server Liberty server.

The following example is a sample server.xml file for an OAuth provider that uses a Derby database store:

```

<server>

 <featureManager>
 <feature>oauth-2.0</feature>
 <feature>ssl-1.0</feature>
 <feature>jdbc-4.0</feature>
 <feature>jndi-1.0</feature>
 </featureManager>

 <keyStore password="keyspass" />

 <oauth-roles>
 <authenticated>
 <user>testuser</user>
 </authenticated>
 </oauth-roles>

 <oauthProvider id="OAuthConfigDerby" filter="request-uri%ssodemo"
 oauthOnly="false">
 <databaseStore dataSourceRef="OAuthDerbyDataSource" />
 </oauthProvider>

 <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib" />

 <library id="DerbyLib" filesetRef="DerbyFileset" />

 <fileset id="DerbyFileset" dir="{DERBY_JDBC_DRIVER_PATH}"
 includes="derby.jar" />

```

```

<dataSource id="OAuthDerbyDataSource" jndiName="jdbc/OAuth2DB"
 jdbcDriverRef="DerbyEmbedded">
 <properties.derby.embedded databaseName="D:\oauth2db"
 createDatabase="create" />
</dataSource>

<webAppSecurity allowFailOverToBasicAuth="true" />

<basicRegistry id="basic" realm="BasicRealm">
 <user name="testuser" password="testuserpwd" />
</basicRegistry>
</server>

```

**Note:** **8.5.5.2** The Componentid must be the same as the id of the oauthProvider element in the server.xml file.

### IBM DB2 for persistent OAuth services:

IBM DB2 can be used for persistent OAuth services. For convenience and reference purposes, this topic documents the steps you need to configure DB2 for OAuth persistent service.

To configure DB2 for persistent OAuth services, complete the following steps:

1. Create a database and tables.

Edit and run the following SQL statement to create an OAuth database and table:

```

-- Change oauth2db to the name you want for the database

CREATE DATABASE oauth2db USING CODESET UTF8 TERRITORY US;
CONNECT TO oauth2db;

---- CREATE TABLES ----
CREATE TABLE OAuthDBSchema.OAUTH20CACHE
(
 LOOKUPKEY VARCHAR(256) NOT NULL,
 UNIQUEID VARCHAR(128) NOT NULL,
 COMPONENTID VARCHAR(256) NOT NULL,
 TYPE VARCHAR(64) NOT NULL,
 SUBTYPE VARCHAR(64),
 CREATEDAT BIGINT,
 LIFETIME INT,
 EXPIRES BIGINT,
 TOKENSTRING VARCHAR(2048) NOT NULL,
 CLIENTID VARCHAR(64) NOT NULL,
 USERNAME VARCHAR(64) NOT NULL,
 SCOPE VARCHAR(512) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 STATEID VARCHAR(64) NOT NULL
 EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}')
);

CREATE TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
(
 COMPONENTID VARCHAR(256) NOT NULL,
 CLIENTID VARCHAR(256) NOT NULL,
 CLIENTSECRET VARCHAR(256),
 DISPLAYNAME VARCHAR(256) NOT NULL,
 REDIRECTURI VARCHAR(2048),
 ENABLED INT
 CLIENTMETADATA CLOB NOT NULL DEFAULT '{}')
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE (

```

```

CLIENTID VARCHAR(256) NOT NULL,
USERID VARCHAR(256),
PROVIDERID VARCHAR(256) NOT NULL,
SCOPE VARCHAR(1024) NOT NULL,
EXPIRES BIGINT,
EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'}
);

---- ADD CONSTRAINTS ----
ALTER TABLE OAuthDBSchema.OAUTH2OCACHE
 ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH2OCLIENTCONFIG
 ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

---- CREATE INDEXES ----
CREATE INDEX OAUTH2OCACHE_EXPIRES ON OAUTHDBSCHEMA.OAUTH2OCACHE (EXPIRES ASC);

---- GRANT PRIVILEGES ----
---- UNCOMMENT THE FOLLOWING IF YOU USE AN ACCOUNT OTHER THAN ADMINISTRATOR FOR DB ACCESS ----

-- Change dbuser to the account you want to use to access your database
-- GRANT ALL ON OAuthDBSchema.OAUTH2OCACHE TO USER dbuser;
-- GRANT ALL ON OAuthDBSchema.OAUTH2OCLIENTCONFIG TO USER dbuser;

---- END OF GRANT PRIVILIGES ----

DISCONNECT CURRENT;

```

The default DB2 listening port is 50000. If you want to find it, run the following command and find the value of the SVCENAME parameter. If it is a number, then it is the port number. If it is a name, look for the name in the /etc/services file or the Windows equivalent if you are using Windows.

```

Linux/Unix: db2 get dbm cfg | grep SVCENAME
Windows: db2 get dbm cfg | findstr SVCENAME

```

You can create a database and tables in DB2 by running the following statement:

```
db2 -tvf createTables.sql
```

## 2. Configure the WebSphere Application Server Liberty server.

The following example is a sample server.xml file for an OAuth provider that uses a DB2 store:

```

<server>
 <featureManager>
 <feature>oauth-2.0</feature>
 <feature>ssl-1.0</feature>
 <feature>jdbc-4.0</feature>
 <feature>jndi-1.0</feature>
 </featureManager>

 <keyStore password="keyspass" />

 <oauth-roles>
 <authenticated>
 <user>testuser</user>
 </authenticated>
 </oauth-roles>

 <oauthProvider id="DB0Auth20Provider" oauthOnly="true"
 filter="request-url%=AnnuityOAuthWeb/index.jsp">
 <databaseStore dataSourceRef="OAUTH2DBDS" />
 </oauthProvider>

 <jdbcDriver id="db2Universal" libraryRef="DB2JCC4LIB" />

```

```

<library apiTypeVisibility="spec,ibm-api,third-party" filesetRef="db2jcc4"
 id="DB2JCC4LIB" />

<fileset dir="${shared.resource.dir}/db2" id="db2jcc4"
 includes="db2jcc4.jar db2jcc_license_cu.jar" />

<dataStore id="OAUTH2DBDS" jdbcDriverRef="db2Universal"
 jndiName="jdbc/oauthProvider">
 <properties.db2.jcc databaseName="OAUTH2DB" driverType="4"
 user="bob" password="abcdefg"
 portNumber="50000"
 serverName="db2.server.mycompany.com" />
</dataStore>

<webAppSecurity allowFailOverToBasicAuth="true" />

<basicRegistry id="basic" realm="BasicRealm">
 <user name="testuser" password="testuserpwd" />
</basicRegistry>
</server>

```

The following example adds a client to DB2:

```

INSERT INTO OAuthDBSchema.OAUTH2OCLIENTCONFIG
(
 COMPONENTID,
 CLIENTID,
 CLIENTSECRET,
 DISPLAYNAME,
 REDIRECTURI,
 ENABLED
)
VALUES
(
 'DBOAuth20Provider',
 'key',
 'secret',
 'My Client',
 'https://localhost:9443/oauth/redirect.jsp',
 1
)

```

**Note:** **8.5.5.2** The Componentid must be the same as the id of the oauthProvider element in the server.xml file.

---

## Configuring Common Secure Interoperability version 2 (CSIv2) in Liberty



**8.5.5.6**

Liberty supports CSIv2 security at various levels such as the message authentication (authentication layer), identity assertion (attribute layer), and client certificate authentication (transport layer). Using the CSIv2 feature, you can specify the type of authentication for both inbound and outbound requests to downstream servers. CSIv2 features are enabled automatically when the appSecurity-2.0 and ejbRemote-3.2 features are configured in the server.xml file. You can configure CSIv2 in Liberty to enable interoperability between Java Platform, Enterprise Edition vendors.

## Procedure

The following is the default configuration that is used without having to specify it in the `server.xml` file when the `appSecurity-2.0` and `ejbRemote-3.2` features are configured.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

You can change each of the layers in `serverPolicy.csiv2` and in `clientPolicy.csiv2` for customizing the inbound and outbound CSIV2 settings.

## Configuring inbound CSIV2 in Liberty



8.5.5.6

Common Secure Interoperability, version 2 (CSIV2) feature inbound configuration determines the type of accepted authentication for inbound requests. The CSIV2 feature is enabled automatically when the `appSecurity-2.0` and `ejbRemote-3.2` features are configured in the `server.xml` file.

### Before you begin

Understand the CSIV2 concepts, see [Common Secure Interoperability version 2 \(CSIV2\)](#) for further information.

### About this task

The following layers of security are available for inbound requests:

#### Procedure

1. Configure the inbound CSIV2 attribute layer.
2. Configure the inbound CSIV2 authentication layer.
3. Configure the inbound CSIV2 transport layer.

### Configuring inbound CSIV2 attribute layer



8.5.5.6

You can configure a Liberty server to claim support for identity assertion for inbound CSIV2 requests.

### About this task

The inbound CSIV2 attribute layer for a Liberty server has identity assertion that is disabled by default. The server supports Anonymous, Principal Name, X509 Certificate Chain, and Distinguished Name identity assertions from an upstream server that is acting as a client after the identity assertion is enabled through the `identityAssertionEnabled` attribute. You can use the `identityAssertionTypes` attribute to

specify the identity token types that the server supports. The `trustedIdentities` attribute can be used to specify the identity of the trusted upstream servers that are able to assert an identity to this server.

**CAUTION:**

Ensure that only trusted entities communicate with the server if presumed trust is set.

**Procedure**

1. Add the `appSecurity-2.0` and `ejbRemote-3.2` features in the `server.xml` file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>
```

The following is the default configuration without having to specify it in the `server.xml` file.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

2. Optional: If you need to change the default inbound attribute layer configuration, then add an `<orb>` element in the `server.xml` file as follows or add the `attributeLayer` element to an existing one. Replace the sample values in the example with your values.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="true"/>
 </layers>
 </serverPolicy.csiv2>
</orb>
```

**Note:** The ID value `defaultOrb` in the `<orb>` element is predefined and cannot be modified.

3. Set the `trustedIdentities` attribute by changing the example values to the `trustedIdentity` of each of the upstream servers. The pipe character (`|`) must be used to separate the values when there are more than one asserting client.

```
<attributeLayer identityAssertionEnabled="true" trustedIdentities="yourAssertingUpstreamServer|anotherAssertingUpstreamServer"/>
```

4. Alternative: Instead of setting a named value for the `trustedIdentities` in step 2, you can set the `trustedIdentities` attribute with the character (`*`) to indicate that the server supports presumed trust. With presumed trust, any upstream server is able to assert an identity and must be used only when the upstream servers can be limited to a set of trusted servers. Therefore, use this value with caution.

```
<attributeLayer identityAssertionEnabled="true" trustedIdentities="*/>
```

5. When an upstream server that sends a certificate chain is trusted, add the issuer distinguished name of the certificate chain to the `trustedIdentities` attribute. For example,

```
<attributeLayer identityAssertionEnabled="true" trustedIdentities="CN=localhost,0=ibm,C=us"/>
```

6. Optional: If you need to change the default identity assertion token types that are supported by the server, then add the `identityAssertionTypes` attribute to the `attributeLayer` element in the `server.xml` file and specify a comma-separated list of values. The valid values are `ITTAnonymous`, `ITTPrincipalName`, `ITTX509CertChain`, and `ITTDistinguishedName`. For example, you can configure the

server to support identity assertions with X509 Certificate Chains or Distinguished Names. Replace the sample values in the example with your values.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="true" identityAssertionTypes="ITTX509CertChain, ITTDistingu
 </layers>
 </serverPolicy.csiv2>
</orb>
```

**Note:** The upstream server identity is obtained from the security information that the server sent in either the authentication layer or the transport layer. The authentication layer identity takes precedence over the transport identity, and the transport identity is used if no security information is sent at the authentication layer. For sample syntax and more information about `attributeLayer` and `transportLayer` elements, see [Configuring inbound CSiv2 authentication layer](#) and [Configuring inbound CSiv2 transport layer](#).

Omitting a layer uses the default values for that layer.

## Results

Your inbound CSiv2 attribute layer is now configured for identity assertion.

## Configuring inbound CSiv2 authentication layer



8.5.5.6

You can configure a Liberty server to use a specific authentication mechanism for inbound CSiv2 requests.

### About this task

The inbound CSiv2 authentication layer for a Liberty server is enabled with the support for the LTPA and GSSUP authentication mechanisms by default. The `establishTrustInClient` association option of the authentication layer is set to `Required` by default to indicate that the authentication mechanisms specified are required. When you are using the LTPA mechanism, ensure that the communicating Liberty servers and other servers share the same LTPA keys.

### Procedure

1. Add the `appSecurity-2.0` and `ejbRemote-3.2` features in the `server.xml` file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>
```

The following example shows the default configuration without having to specify it in the `server.xml` file.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

```

 <transportLayer/>
 </layers>
</clientPolicy.csiv2>
</orb>

```

- Optional: If you need to change the default inbound authentication layer configuration, then add an <orb> element in the server.xml file as follows or add the authenticationLayer element to an existing one. Replace the sample values in the example with your values.

```

<orb id="default0rb">
 <serverPolicy.csiv2>
 <layers>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 </layers>
 </serverPolicy.csiv2>
</orb>

```

**Note:** The ID value default0rb in the <orb> element is predefined and cannot be modified.

- Optional: Set the mechanisms attribute to LTPA or GSSUP to use either LTPA or GSSUP (user name and password) only as the authentication mechanism.

```

<authenticationLayer mechanisms="LTPA" establishTrustInClient="Supported"/>
or
<authenticationLayer mechanisms="GSSUP" establishTrustInClient="Supported"/>

```

- Optional: Set the establishTrustInClient attribute to Required, Supported, or Never to indicate that the server requires, supports (optional), or never claims authentication with the specified mechanisms.

#### Notes:

- When the establishTrustInClient attribute is set to Required, only clients that either require or support compatible (at least one) authentication mechanisms are able to send a security context to the server.
- When the establishTrustInClient attribute is set to Supported, a client might choose whether to send the authentication information in the authentication layer.
- When the establishTrustInClient attribute is set to Never, the inbound CSIV2 authentication layer is disabled and at least one other CSIV2 layer must be enabled to authenticate.

Omitting a layer uses the default values for that layer.

For more information about the attributeLayer and transportLayer elements, see [Configuring inbound CSIV2 attribute layer](#) and [Configuring inbound CSIV2 transport layer](#).

## Results

Your inbound CSIV2 authentication layer is now configured.

## Configuring inbound CSIV2 transport layer



8.5.5.6

You can configure a Liberty server to claim support for client certificate authentication for inbound CSIV2 requests.

### About this task

The inbound CSIV2 transport layer for a Liberty server has client certificate authentication that is disabled by default. You can configure the transportLayer to specify the SSL configuration to use. You can configure the SSL element to either support or require the client certificate authentication. The certificate that is received is authenticated against the server user registry, and its identity is only used if no other form of authentication was sent in the CSIV2 request, like an identity assertion in the attribute layer or an authentication token in the authentication layer.



When you use the client certificate authentication, ensure that SSL is supported by the server.

## Procedure

1. Add the appSecurity-2.0 and ejbRemote-3.2 features in the server.xml file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>
```

The following example is the default configuration without having to specify it in the server.xml file.

```
<orb id="default0rb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

2. Configure SSL support as described in the “Enabling SSL communication in Liberty” on page 1152 page.
3. Configure the SSL element to use clientAuthentication or clientAuthenticationSupported. For example,

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthentication="true" />
```

or

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
```

- If you specify clientAuthentication="true", then the server requests that a client sends a certificate. However, if the client does not have a certificate or if the certificate is not trusted by the server, then the handshake does not succeed.
  - If you specify clientAuthenticationSupported="true", then the server requests that a client sends a certificate. However, if the client does not have a certificate or if the certificate is not trusted by the server, then the handshake might still succeed.
  - If you do not specify either clientAuthentication or clientAuthenticationSupported, or if you specify clientAuthentication="false" or clientAuthenticationSupported="false", then the server does not request the client to send a certificate during the handshake.
4. Optional: If you need to change the default inbound transport layer configuration, then add an <orb> element in the server.xml file as follows or add the transportLayer element to an existing one. Replace the sample values in the example with your values.

```
<orb id="default0rb">
 <serverPolicy.csiv2>
 <layers>
 <transportLayer sslRef="defaultSSLConfig"/>
 </layers>
 </serverPolicy.csiv2>
</orb>
```

**Note:** The ID value default0rb in the <orb> element is predefined and cannot be modified.

5. Make sure the server trusts any client certificates that are used.

6. Make sure any client certificates that are used for client authentication are mapped to a user identity in your registry.
  - For the basic registry, the user identity is the common name (CN) from the distinguished name (DN) of the certificate.
  - For a Lightweight Directory Access Protocol (LDAP) registry, the DN from the client certificate must be in the LDAP registry.

Omitting a layer uses the default values for that layer. For more information about `attributeLayer` and `authenticationLayer` elements, see [Configuring inbound CSIV2 attribute layer](#) and [Configuring inbound CSIV2 authentication layer](#).

## Results

Your inbound CSIV2 transport layer is now configured for client certificate authentication.

## Configuring outbound CSIV2 in Liberty



8.5.5.6

The Common Secure Interoperability, version 2 (CSIV2) feature outbound configuration determines the type of authentication information sent for outbound requests. The CSIV2 feature is enabled automatically when the `appSecurity-2.0` and `ejbRemote-3.2` features are configured in the `server.xml` file.

### Before you begin

Understand the CSIV2 concepts, see [Common Secure Interoperability version 2 \(CSIV2\)](#) for further information.

### About this task

The following layers of security are available for outbound requests:

#### Procedure

1. Configure the outbound CSIV2 attribute layer.
2. Configure the outbound CSIV2 authentication layer.
3. Configure the outbound CSIV2 transport layer.

### Configuring outbound CSIV2 attribute layer



8.5.5.6

You can configure a Liberty server to perform identity assertions for outbound CSIV2 requests.

### About this task

Identity assertion is disabled by default in the outbound CSIV2 attribute layer for a Liberty server. The server that is acting as a client supports sending the Principal Name and Anonymous identity assertions to a downstream server after the identity assertion is enabled through the `identityAssertionEnabled` attribute. You can use the `identityAssertionTypes` attribute to specify more or different identity token types that the server supports for outbound requests. The `trustedIdentity` and `trustedPassword` attributes can be used to specify the identity of the client to be verified for trust by the downstream server when the authentication layer mechanism is GSSUP. The `trustedIdentity` attribute can be set without a `trustedPassword` if the authentication mechanism in the authentication layer is LTPA. You must also configure the upstream server along with enabling the identity assertion so that the client can assert an identity.

## Procedure

1. Add the appSecurity-2.0 and ejbRemote-3.2 features in the server.xml file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>
```

The following example is the default configuration without having to specify it in the server.xml file.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

2. Optional: If you need to change the default outbound attribute layer configuration, then add an <orb> element in the server.xml file as follows or add the attributeLayer element to an existing one. Replace the sample values in the example with your values.

```
<orb id="defaultOrb">
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="true"/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

**Note:** The ID value defaultOrb in the <orb> element is predefined and cannot be modified.

3. Specify the upstream server identity for trust validation by the downstream server. The trustedIdentity specified must exist in the user registry of the target server.
  - When you are using the GSSUP mechanism in the authentication layer, you must set the trustedIdentity and trustedPassword attributes by changing the example values to the identity and password of the upstream server that is acting as a client.

```
<attributeLayer identityAssertionEnabled="true" trustedIdentity="yourTrustedId" trustedPassword="yourTrustedIdPwd"/>
```

Encode the password within the configuration. You can get the encoded value by using the securityUtility encode command.
  - When you are using the LTPA mechanism in the authentication layer, you must set the trustedIdentity attribute by changing the example value to the identity of the upstream server that is acting as a client.

```
<attributeLayer identityAssertionEnabled="true" trustedIdentity="yourTrustedId"/>
```
4. Optional: If you need to change the default identity assertion token types that are supported by the server, then add the identityAssertionTypes attribute to the attributeLayer element in the server.xml file and specify a comma-separated list of values. The valid values are ITTAnonymous, ITTPrincipalName, ITTX509CertChain, and ITTDistinguishedName. For example, you can configure the server to support identity assertions with X509 Certificate Chains or Distinguished Names. Replace the sample values in the example with your values.

```
<orb id="defaultOrb">
 <clientPolicy.csiv2>
 <layers>
```

```

 <attributeLayer identityAssertionEnabled="true" identityAssertionTypes="ITTX509CertChain, ITTDistinguish
 </layers>
</clientPolicy.csiv2>
</orb>

```

#### Notes:

- If both LTPA and GSSUP are configured in the authentication layer and the downstream server supports LTPA, then LTPA takes precedence over GSSUP.
- If both LTPA and GSSUP are configured in the authentication layer and the downstream server supports only GSSUP, then GSSUP is used and the trustedIdentity and trustedPassword attributes must be specified.
- The trustedIdentity attribute is not required if you are using the transport certificate chain to identify the server to the downstream server. (The identityAssertionEnabled attribute is set to true and establishTrustInClient is set to Never in the authenticationLayer).
- Omitting a layer uses the default values for that layer.

For more information about authenticationLayer and transportLayer elements, see [Configuring outbound CSiv2 authentication layer](#) and [Configuring outbound CSiv2 transport layer](#).

## Results

Your outbound CSiv2 attribute layer is now configured for identity assertion.

## Configuring outbound CSiv2 authentication layer



8.5.5.6

You can configure a Liberty server to use specific authentication mechanisms for outbound CSiv2 requests.

### About this task

The outbound CSiv2 authentication layer for a Liberty server is enabled with support for the LTPA and GSSUP authentication mechanisms by default. The establishTrustInClient association option of the authentication layer is set to Supported by default to indicate that the authentication mechanisms specified are supported and optional.

When the LTPA mechanism is used, ensure that the communicating Liberty servers and other servers share the same LTPA keys.

### Procedure

1. Add the appSecurity-2.0 and ejbRemote-3.2 features in the server.xml file.

```

<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>

```

The following example is the default configuration without having to specify it in the server.xml file.

```

<orb id="default0rb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
</clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>

```

```

 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
</clientPolicy.csiv2>
</orb>

```

- Optional: If you need to change the default outbound authentication layer configuration, then add an `<orb>` element in the `server.xml` file as follows or add the `authenticationLayer` element to an existing one. Replace the sample values in the example with your values.

```

<orb id="default0rb">
 <clientPolicy.csiv2>
 <layers>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 </layers>
 </clientPolicy.csiv2>
</orb>

```

**Note:** The ID value `default0rb` in the `<orb>` element is predefined and cannot be modified.

- Optional: Set the `mechanisms` attribute to `LTPA` or `GSSUP` to use either LTPA or GSSUP (user name and password) only as the authentication mechanism.

```
<authenticationLayer mechanisms="LTPA" establishTrustInClient="Supported"/>
```

or

```
<authenticationLayer mechanisms="GSSUP" establishTrustInClient="Supported"/>
```

- Optional: Set the `establishTrustInClient` attribute to `Required`, `Supported`, or `Never` to indicate that the server that is acting as a client requires, supports(optional), or never performs authentication with the specified mechanisms.

#### Notes:

- When the `establishTrustInClient` attribute is set to `Required`, the client is able to send an authentication token of one of the specified mechanisms only to servers that either require or support the same authentication mechanisms.
- When the `establishTrustInClient` attribute is set to `Supported`, the client might choose whether to send the authentication information in the authentication layer. If the downstream server is configured with `Supported` or `Required`, then the client sends a compatible authentication token.
- When the `establishTrustInClient` attribute is set to `Never`, the outbound CSIV2 authentication layer is disabled and at least one other CSIV2 layer must be enabled to authenticate to the downstream server.
- Omitting a layer uses the default values for that layer.

For more information about the `attributeLayer` and `transportLayer` elements, see [Configuring outbound CSIV2 attribute layer](#) and [Configuring outbound CSIV2 transport layer](#). For an example of a programmatic login when using GSSUP as the authentication mechanism, see [Example: Using the WSLogin configuration to create a basic authentication subject](#).

## Results

Your outbound CSIV2 authentication layer is now configured.

## Configuring outbound CSIV2 transport layer



8.5.5.6

You can configure a Liberty server to perform client certificate authentication for outbound CSIV2 requests.

## About this task

The client certificate authentication of the outbound CSiv2 transport layer for a Liberty server is disabled by default. You can configure the transportLayer to specify the SSL configuration to use.

You can configure the SSL element to support client certificate authentication or require it. The certificate sent to the downstream server is authenticated against the downstream server user registry and its identity is only used if no other form of authentication is sent in the CSiv2 request, like an identity assertion in the attribute layer or an authentication token in the authentication layer.

When the client certificate authentication is used, ensure that SSL is supported by this server.

## Procedure

1. Add the appSecurity-2.0 and ejbRemote-3.2 features in the server.xml file.

```
<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>ejbRemote-3.2</feature>
</featureManager>
```

The following example is the default configuration without having to specify it in the server.xml file.

```
<orb id="defaultOrb">
 <serverPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Required"/>
 <transportLayer/>
 </layers>
 </serverPolicy.csiv2>
 <clientPolicy.csiv2>
 <layers>
 <attributeLayer identityAssertionEnabled="false"/>
 <authenticationLayer mechanisms="LTPA,GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

2. Configure SSL support as described in Enabling SSL communication for Liberty.
3. Optional: Configure the SSL element to use clientAuthentication or clientAuthenticationSupported. For example,

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthentication="true" />
```

or

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
```

4. Optional: If you need to change the default outbound transport layer configuration, then add an <orb> element in the server.xml file as follows or add the transportLayer element to an existing one. Replace the sample values in the example with your values.

```
<orb id="defaultOrb">
 <clientPolicy.csiv2>
 <layers>
 <transportLayer sslRef="defaultSSLConfig"/>
 </layers>
 </clientPolicy.csiv2>
</orb>
```

**Note:** The id value *defaultOrb* in the <orb> element is predefined and cannot be modified.

5. Make sure the downstream server trusts any client certificates that are sent from this server.

6. Make sure that any client certificates used for client authentication are mapped to a user identity in the downstream server user registry.
  - For the basic registry, the user identity is the common name (CN) from the distinguished name (DN) of the certificate.
  - For a Lightweight Directory Access Protocol (LDAP) registry, the DN from the client certificate must be in the LDAP registry.

**Notes:**

- When the `clientAuthentication` attribute is set to `true` in the `<ssl>` element, the client sends a client certificate only to servers that either require or support the client certificate authentication.
- When the `clientAuthenticationSupported` attribute is set to `true` in the `<ssl>` element, the client might choose whether to send a client certificate based on the `<ssl>` element configuration that is used by the downstream server.
- When the `clientAuthentication` and `clientAuthenticationSupported` attributes are not set in the `<ssl>` element, then the server that is acting as a client is not enabled with the client certificate authentication.

Omitting a layer uses the default values for that layer. For more information about the `attributeLayer` and `authenticationLayer` elements, see [Configuring outbound CSIV2 attribute layer](#) and [Configuring outbound CSIV2 authentication layer](#).

## Results

Your outbound CSIV2 transport layer is now configured for client certificate authentication.

---

## Configuring security for the Liberty application client container and its applications



8.5.5.6

Configure security on the Liberty application client container and its applications to ensure communications between the client and server are secure. You can also configure security to ensure that credentials of clients flow to the server.

### Enabling SSL communication for the Liberty application client container



8.5.5.6

The Liberty application client container might require some SSL configuration for the client container to communicate with a server. Configuring SSL for the application client container requires the use of the same SSL feature, `ssl-1.0`, that the server requires for SSL enablement. The configuration elements and attributes are the same for the application client as for the server; however, for the application client container, these values are specified in the `client.xml` file.

### About this task

The SSL handshake is a series of messages that are exchanged over the SSL protocol between a client and a server to negotiate for connection-specific protection. To enable SSL for the Liberty application client container the SSL feature, `ssl-1.0`, must include the minimal information that is needed to form an SSL configuration that is used by the client. The minimal information that is required to form an SSL configuration is a keystore and a password.

You can use the securityUtility **createSSLCertificatecommand** to create the keystore of the client and to provide you with information about the configuration. Using the tool is optional, because you can also create a keystore and the associated configuration for other customer-defined purposes.

## Procedure

Add the keystore element to the `client.xml` file. The `id` attribute must be `defaultKeyStore` and the `password` attribute contains the keystore password. The password can be entered in clear text or encoded. Use the securityUtility `encode` option to encode the password.

```
<keyStore id="defaultKeyStore" password="yourPassword" />
```

This is the minimum configuration that is needed to create an SSL configuration. In this configuration, the client creates the keystore and certificate if it does not exist during SSL initialization. The password that is provided must be at least 6 characters long. JKS is the default keystore type and the default keystore is called `key.jks` and these defaults are in the `<client home>/resources/security` directory. The client will create the `defaultKeyStore` the first time it starts when using the previous configuration, however having the client create the default certificate comes with a performance cost. To avoid the performance cost it is recommended that the **securityUtility createSSLCertificate** command be used to create the default keystore used in the `defaultKeyStore` configuration. If you need a custom SSL configuration, see SSL configuration attributes.

### Accepting signer certificates

If the client does not have an established trust relationship with the server the communication with the client prompts the user and asks if they accept the certificate from the server. If the user responds with yes, the certificate is taken and stored in the client keystore configuration and the command proceeds. If the user specifies no then there is no trust that is established and the call ends in an error.

Example of what the prompt looks like:

```
*** SSL SIGNER EXCHANGE PROMPT ***
```

```
The SSL signer from target host is not found in trust store C:/liberty/workspace/build.image/wlp/usr/clients/myTest
```

Here is the signer information (verify the digest value matches what is displayed at the server):

```
Subject DN: CN=localhost, O=ibm, C=us
Issuer DN: CN=localhost, O=ibm, C=us
Serial number: 1327582458
Expires: Sun Jan 04 06:54:18 CST 2099
SHA-1 Digest: 00:6F:25:F1:78:5D:EB:00:B1:E2:99:DB:E8:D7:DF:3B:F8:E0:20:9A
Add signer to the trust store now? (y/n)
```

You might receive the following error message if the user specifies no when asked to add the signer to the truststore:

```
[ERROR] CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN CN=localhost, O=ibm, C=us
sent from the target host. The signer might need to be added to local trust store C:/liberty/workspace/build.image
java.security.cert.CertPathValidatorException: The certificate issued by SubjectDN CN=localhost, O=ibm, C=us
java.security.cert.CertPathValidatorException: Certificate chaining error
throw able: javax.net.ssl.SSLHandshakeException: java.security.cert.CertificateException: PKIX path building failed
: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause is:

 java.security.cert.CertPathValidatorException: The certificate issued by
SubjectDN CN=localhost, O=ibm, C=us is not trusted; internal cause is:
 java.security.cert.CertPathValidatorException: Certificate chaining error
```

### Auto accept signer certificate

If the client does not want to be prompted for the signer certificate and chooses to accept the server signer certificate without examining the certificate, the user can provide the `-autoAcceptSigner` flag to the client container command line.

```
client run client_name --autoAcceptSigner
```

### Client authentication

If the client is communicating with a server that has client authentication that is enabled, then the



server needs to trust the client as well as the client trusting the server. The client must have a key, and personal certificate, in its keystore. If you use the `securityUtility createSSLCertificate` command, the keystore contains a personal certificate. The server that the client application container is communicating with must trust the client, so the signer from the client needs to be added to the truststore of the server. You can use the java tool, `keytool`, to extract the signer from the keystore of the application client container and add the certificate from the client to the truststore of the server.

## Configuring a JAAS programmatic login on the Liberty application client container



8.5.5.6

The Liberty application client container can be configured to use a JAAS programmatic login.

### Before you begin

Review the different ways of authenticating users on the application client container, and decide whether the programmatic login option is best for your environment. For further details, see Authentication on the Liberty application client container.

### About this task

A programmatic login is a type of form login that supports application presentation login forms for authentication. This approach requires the application developer to collect the user's credentials and authenticate that user. This method takes advantage of the JAAS framework to send a user's credentials to the server for authentication. The JAAS framework consists of creating a login context by specifying a JAAS login configuration and by using a callback handler to gather the user's credentials. When a subject is obtained from the login context, you can use a Liberty security API to set that Subject on the thread, and it is used for your outbound call to the server.

The JAAS login configuration specifies how and which login modules are used for authentication. Here are the JAAS login configurations that are provided by Liberty on the client:

- **WSLogin JAAS login configuration:** A generic JAAS login configuration that a Liberty application client container application can use to perform authentication that is based on a user ID and password. However, this configuration does not support the `CallbackHandler` handler that is specified in the deployment descriptor of the client application module.
- **ClientContainer JAAS login configuration:** This JAAS login configuration recognizes the `CallbackHandler` handler that is specified in the client application module's deployment descriptor, if one is specified. If a handler is not specified in the deployment descriptor, then the handler that was specified programmatically is used.

The login modules that are specified by the JAAS login configuration implement a certain authentication technology. A login module can gather credentials from the user by using the `javax.security.auth.callback.CallbackHandler` interface. Liberty provides a non-prompt implementation of the `CallbackHandler` interface, which is called `com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl`. This implementation enables an application developer to specify the credentials directly in the application without having to prompt the user. There are two ways to specify your `CallbackHandler` implementation:

- Specify your implementation programmatically, as an argument to the `javax.security.auth.login.LoginContext` constructor; for example:

```
LoginContext logincontext = new LoginContext("ClientContainer", new WSCallbackHandlerImpl("user", "password"));
```
- Specify your implementation name in the client application module's deployment descriptor (`application-client.xml`); for example:

```
<callbackhandler>com.acme.callbackhandler.WSCallbackHandlerImpl</callbackhandler>
```

**Note:** The WSLLogin login configuration does not recognize the second option of specifying a CallbackHandler handler in the deployment descriptor.

## Procedure

1. Add the appSecurityClient-1.0 feature to your client.xml file.  

```
<feature>appSecurityClient-1.0</feature>
```
2. Configure SSL for your client:
  - a. **Optional:** Use the **securityUtility** command to create an SSL certificate for the client; for example:  

```
securityUtility createSSLCertificate --client=myClient --password=liberty
```
  - b. **Recommended:** Use the **securityUtility** command to generate an xor encoded password. For example, to encode the password liberty:  

```
securityUtility encode liberty
```
  - c. Add a keyStore element to your client.xml file. The following example uses the default SSL configuration:  

```
<keyStore id="defaultKeyStore" password="{xor}MzY90i0rJg==" /> <!-- pwd: liberty -->
```
3. In the application code, create a Subject using the ClientContainer JAAS login configuration and the WSCallbackHandlerImpl callback handler.
  - a. Before the application makes an outbound request, add the following code. Change the userName and userPassword to valid credentials for a user that exists in the user registry of the target server.

```
CallbackHandler wscbh = new WSCallbackHandlerImpl("userName", "userPassword");
LoginContext ctx = null;
try {
 ctx = new LoginContext("ClientContainer", wscbh);
} catch (LoginException le) {
 le.printStackTrace();
}
try {
 ctx.login();
} catch (LoginException le) {
 le.printStackTrace();
}
Subject subject = ctx.getSubject();
```
4. Set the Subject obtained in the previous step on the thread and use that Subject to look up an EJB. Use the WSSubject.doAs, or doAsPrivileged APIs to accomplish this action. The Subject within the com.ibm.websphere.security.auth.WSSubject.doAs or com.ibm.websphere.security.auth.WSSubject.doAsPrivileged code block is used for Java Platform, Enterprise Edition (J2EE) resources authorization checks.

```
WSSubject.doAs(subject, new PrivilegedAction() {
 public Object run() {
 try {
 //Perform EJB lookup and invocation
 } catch (Exception ex) {
 ex.printStackTrace();
 }
 return null;
 }
});
```
5. 8.5.5.7 If Java 2 security is enabled on your client and your application code is calling JAAS or Liberty security APIs, add the necessary Java 2 Security permissions to either the permissions.xml file or the client.xml file of the application. For more details on which Liberty security APIs are protected by Java 2 Security permissions, see Programming Interfaces (APIs). For further details, see Java 2 Security.

## What to do next

As on the server, you can use a custom login module to either make more authentication decisions or add information to the subject to make finer-grained authorization decisions inside your client application. For further details, see [Configuring a JAAS custom login module for the Liberty application client container](#).

## Configuring a JAAS custom login module for the Liberty application client container



8.5.5.6

You can configure the Liberty application client container to use a custom Java Authentication and Authorization Service (JAAS) login module.

### Before you begin

Make sure that you have a JAR file that contains the JAAS custom login module, which implements the `javax.security.auth.spi.LoginModule` interface.

### About this task

You can use a custom login module to either make additional authentication decisions or add information to the subject to make finer-grained authorization decisions inside your application. To configure a JAAS custom login module, complete the following steps.

### Procedure

1. Add the `appSecurityClient-1.0` feature to your `client.xml` file.

```
<feature>appSecurityClient-1.0</feature>
```

2. Create a class `com.sample.CustomLoginModule` that implements the `LoginModule` interface and package it into the `CustomLoginModule.jar` file.

3. Create a `<library>` element that uses a `<fileset>` element that indicates where the `CustomLoginModule.jar` file is. In this example, the file is in the client's configuration directory, and the library id is `customLoginLib`.

```
<library id="customLoginLib">
 <fileset dir="{server.config.dir}" includes="CustomLoginModule.jar"/>
</library>
```

4. Create a `<jaasLoginModule>` element. In this example, the id is `myCustom`.

- a. Configure the custom login module to require a successful authentication by setting the `controlFlag` attribute to `REQUIRED`.
- b. Set the `libraryRef` attribute to `customLoginLib`, the id of the `<library>` element that is configured in the previous step.

```
<jaasLoginModule id="myCustom" className="com.sample.CustomLoginModule" controlFlag="REQUIRED" libraryRef="customLoginLib"/>
```

5. Create a `<jaasLoginContextEntry>` element with the id and name of the system-defined JAAS configuration on the application client container: `ClientContainer`. You can also set this JAAS configuration to `WSLogin`, or your own JAAS configuration. In the `loginModuleRef` attribute, add `proxy`, the id for the proxy login module and `myCustom`, the id of the `jaasLoginModule` element that is created in the previous step.

```
<jaasLoginContextEntry id="ClientContainer" name="ClientContainer"
 loginModuleRef="proxy, myCustom"/>
```

# Configuring Common Secure Interoperability version 2 (CSiv2) in the Liberty application client container



8.5.5.6

The WebSphere Application Server Liberty application client container supports CSiv2 security at various levels such as the message authentication (message layer), and client certificate authentication (transport layer). Using the CSiv2 feature, you can specify the type of authentication for outbound requests to the servers. CSiv2 features are enabled by default. You can configure CSiv2 in the Liberty application client container to enable interoperability between Java Platform, Enterprise Edition vendors.

## Configuring outbound CSiv2 in the Liberty application client container



8.5.5.6

CSiv2 features are enabled by default. You can configure CSiv2 in the Liberty application client container to enable interoperability between Java Platform, Enterprise Edition vendors.

### Before you begin

Understand the CSiv2 concepts, see Common Secure Interoperability version 2 (CSiv2) for further information.

### Configuring the outbound CSiv2 authentication layer in the Liberty application client container:



8.5.5.6

You can configure a Liberty application client container to use specific authentication mechanisms for outbound CSiv2 requests.

### About this task

The outbound CSiv2 authentication layer for a Liberty application client container is enabled with support for the GSSUP authentication mechanism by default. The `establishTrustInClient` association option of the authentication layer is set to `Supported` by default to indicate that the authentication mechanisms specified are supported and optional.

### Procedure

1. Configure the `orb` element in the `client.xml` file as follows or add the `authenticationLayer` element to an existing one, replacing the sample values in the example with your values:

```
<orb id="default0rb">
 <clientPolicy.clientContainerCsiv2>
 <layers>
 <authenticationLayer user="userId" password="{xor}PDC+MTg6Ejo="/>
 </layers>
 </clientPolicy.clientContainerCsiv2>
</orb>
```

**Note:** The `id` value `default0rb` in the `orb` element is predefined and cannot be modified.

**Note:** Hash encoding cannot be used for encrypting the password because the original password cannot be decoded from the hashed value.

The `mechanisms` and `establishTrustInClient` attributes are optional. The only supported value, and the default value, for the `mechanisms` attribute is `GSSUP`.

Without specifying an `<orb>` element, the following configuration is implicit.

```

<orb id="defaultOrb">
 <clientPolicy.clientContainerCsiv2>
 <layers>
 <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.clientContainerCsiv2>
</orb>

```

2. **Optional:** Set the user and password attributes with a valid user ID and password to access the server. By default, a server requires the GSSUP mechanism for inbound connections, meaning that the server must receive a user and password and because of this requirement, the user, and password values are required in the `client.xml` file, unless a programmatic login is implemented by the application.
3. **Optional:** Set the `establishTrustInClient` attribute to Required, Supported (default), or Never for performing authentication with the specified mechanisms. For example,

```

<orb id="defaultOrb">
 <clientPolicy.clientContainerCsiv2>
 <layers>
 <authenticationLayer user="userId" password="{xor}PDC+MTg6Ejo=" establishTrustInClient="Required" />
 </layers>
 </clientPolicy.clientContainerCsiv2>
</orb>

```

**Note:**

- When the `establishTrustInClient` attribute is set to Required, the client is able to send an authentication token of one of the specified mechanisms only to servers that either require or support the same authentication mechanisms.
- When the `establishTrustInClient` attribute is set to Supported (default), the client can choose whether to send the authentication information in the authentication layer. If the server is configured with Supported or Required of the same authentication mechanisms, then the client sends a compatible authentication token.
- When the `establishTrustInClient` attribute is set to Never, the outbound CSIV2 authentication layer is disabled and the CSIV2 transport layer must be enabled to authenticate to the server.

## Results

Your outbound CSIV2 authentication layer is now configured.

## Configuring the outbound CSIV2 transport layer in the Liberty application client container:

8.5.5.6

You can configure the Liberty application client container to perform client certificate authentication for outbound CSIV2 requests.

### About this task

The client certificate authentication of the outbound CSIV2 transport layer for a Liberty application client container is not used by default. You can configure the `transportLayer` to specify the SSL configuration to use.

You can configure the SSL element to support client certificate authentication or require it. The certificate sent to the server is authenticated against the server user registry and its identity is only used if no other form of authentication is sent in the CSIV2 request, like an identity assertion in the attribute layer or an authentication token in the authentication layer.

## Procedure

1. Configure SSL support as described in Enabling SSL communication for the Liberty application client container.
2. Optional: Configure the SSL element to use `clientAuthentication` or `clientAuthenticationSupported`. For example,

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthentication="true" />
```

or

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
 trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
```

3. Configure the `<orb>` element in the `client.xml` file as follows or add the `transportLayer` element to an existing one and replace the sample values in the sample with your values:

```
<orb id="defaultOrb">
 <clientPolicy.clientContainerCsi2>
 <layers>
 <transportLayer sslRef="defaultSSLConfig"/>
 </layers>
 </clientPolicy.clientContainerCsi2>
</orb>
```

Without specifying an `<orb>` element, the following configuration is implicit.

```
<orb id="defaultOrb">
 <clientPolicy.clientContainerCsi2>
 <layers>
 <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Supported"/>
 <transportLayer/>
 </layers>
 </clientPolicy.clientContainerCsi2>
</orb>
```

4. Make sure the server trusts any client certificates that are sent from this server.
  - When the `clientAuthentication` attribute is set to `true` in the `ssl` element, the client sends a client certificate only to servers that either require or support the client certificate authentication.
  - When the `clientAuthenticationSupported` attribute is set to `true` in the `ssl` element, the client might choose whether to send a client certificate based on the `ssl` element configuration used by the server.
  - When the `clientAuthentication` and `clientAuthenticationSupported` attributes are not set in the `ssl` element, the server that is acting as a client is not enabled with the client certificate authentication.

## Results

Your outbound CSIv2 transport layer is now configured for client certificate authentication.

---

## Configuring Java Servlet 3.1 support for security



8.5.5.4

Liberty supports all security updates as defined in the Java Servlet 3.1 specification.

### About this task

Take advantage of the Java Servlet 3.1 features on Liberty.

## Procedure

1. Add the servlet-3.1 feature in the server.xml file:

```
<feature>servlet-3.1</feature>
```

2. Determine which of the following Java Servlet 3.1 functions that you want to use:

- Specify autocomplete=off in the login form.

When you use HTML for a form login page, set the password form field to autocomplete="off" to disable automatically filling in passwords in the web browser. For example:

```
<form method="POST" action="j_security_check">
<input type="text" name="j_username">
<input type="password" name="j_password" autocomplete="off">
</form>
```

- Specify the all authenticate security constraint (\*\*).

The special role name \*\* indicates any authenticated user. When \*\* displays in an authorization constraint, if the user is authenticated, that user has access to the methods that are specified in the constraint. Users do not have to be mapped to this role in the application bindings. For example:

```
<security-constraint id="SecurityConstraint_1">
 <web-resource-collection id="WebResourceCollection_1">
 <web-resource-name>Protected with ** role</web-resource-name>
 <url-pattern>/AnyAuthSecurityConstraint</url-pattern>
 <http-method>GET</http-method>
 <http-method>POST</http-method>
 </web-resource-collection>
 <auth-constraint id="AuthConstraint_1">
 <role-name>**</role-name>
 </auth-constraint>
</security-constraint>
```

When the isUserInRole() method is called with a role name of \*\*, isUserInRole() returns true if the user is authenticated. If \*\* is a defined role in the configuration in a security-role, it is not treated like the special any authenticated user role. The user must be mapped to that role in application bindings for isUserInRole to return true.

- Specify the deny-uncovered-http-methods flag in web.xml files.

If the deny-uncovered-http-methods element is specified within the web.xml file, the container denies any uncovered HTTP methods that are not enumerated within the combined security constraint for a URL pattern that is the best match for the request URL. A 403 (SC\_FORBIDDEN) status code is returned. For example:

```
<servlet-mapping id="ServletMapping_1">
 <servlet-name>MyServlet</servlet-name>
 <url-pattern>/MyURLPattern</url-pattern>
</servlet-mapping>
```

```
<deny-uncovered-http-methods/>
```

```
<!-- SECURITY CONSTRAINTS -->
```

```
<security-constraint id="SecurityConstraint_1">
 <web-resource-collection id="WebResourceCollection_1">
 <web-resource-name>Protected with Employee or Manager roles</web-resource-name>
 <url-pattern>/MyURLPattern</url-pattern>
 <http-method>GET</http-method>
 <http-method>POST</http-method>
 </web-resource-collection>
 <auth-constraint id="AuthConstraint_1">
 <role-name>Employee</role-name>
 <role-name>Manager</role-name>
 </auth-constraint>
</security-constraint>
```

If the deny-uncovered-http-methods element is specified in the web.xml file, a message is logged in the messages.log file for each URL pattern in each servlet, indicating the uncovered methods with a note that those uncovered methods are unprotected and not accessible. For example:

For URL `MyURLPattern` in servlet `MyServlet`, the following HTTP methods are uncovered, and not accessible: `DELETE OPTIO`  
If the `deny-uncovered-http-methods` element is not specified in the `web.xml` file, a message is logged in the `messages.log` file for each URL pattern in each servlet, indicating the uncovered methods with a note that those uncovered methods are unprotected and accessible. For example:

For URL `MyURLPattern` in servlet `MyServlet`, the following HTTP methods are uncovered, and accessible: `DELETE OPTIONS H`

## Results

You have now secured your application.

---

## Configuring secure JMX connection to Liberty

You can access the secured Java Management Extensions (JMX) connectors on Liberty by using SSL. The secured JMX connection is enabled by the Liberty feature `restConnector-1.0`.

### About this task

The REST connector is enabled through the Liberty feature `restConnector-1.0`. Remote access through the REST connector is protected by a single administrator role. In addition, SSL is required to keep the communication confidential. The `restConnector-1.0` feature already includes the `ssl-1.0` feature.

**Note:** An application deployed on Liberty has unrestricted access to its `MBeanServer` directory.

The following section describes how to configure and access the REST connector on Liberty.

### Procedure

1. Enable the REST connector using the following code in the `server.xml` file.

```
<featureManager>
 <feature>restConnector-1.0</feature>
</featureManager>
```

2. Configure SSL certificates in the `server.xml` file.

Ensure that the CN value of the certificate's subjectDN is the host name of the machine where the server is running, and that the truststore contains the certificate of the server in the `jConsole` connection.

3. Configure a user or group to the administrator role in the `server.xml` file.
  - Map to the administrator role for Liberty
4. Access the REST connector.

**8.5.5.4** You can access a Liberty REST connector from a Java client or directly through an HTTPS call. A Java client uses the client-side of the connector, which is in `wlp/clients/restConnector.jar` and implements the `javax.management.MBeanServerConnection` interface. HTTPS calls use the server-side of the connector. As to HTTPS calls on the server-side, any programming language that can make HTTPS calls, such as C++, JavaScript, curl, Ruby, and Perl, can use the REST APIs. The REST APIs contain endpoints for management (JMX), file transfer, collective routing and collective deployment.

- Access the REST connector from a JMX client application or by using the `jConsole` tool provided in the Java SDK. Use `-J` flags to pass the system properties as Java options and set the class path to include the connector class files. The connector class files are packed in the `clients/restConnector.jar` file.

- Use the following properties for SSL certificates:

```
-J-Djavax.net.ssl.trustStore=<location of your client trust store>
-J-Djavax.net.ssl.trustStorePassword=<password for the trust store>
-J-Djavax.net.ssl.trustStoreType=<type of trust store>
```

The following example shows the `jConsole` tool being used with SSL configurations:



```
jconsole -J-Djava.class.path=%JAVA_HOME%/lib/jconsole.jar;
 %JAVA_HOME%/lib/tools.jar;
 %WLP_HOME%/clients/restConnector.jar
 -J-Djavax.net.ssl.trustStore=key.jks
 -J-Djavax.net.ssl.trustStorePassword=Liberty
 -J-Djavax.net.ssl.trustStoreType=jks
```

After the jConsole starts, select **Remote Process**, and enter the JMX service URL:  
 service:jmx:rest://<host>:<port>/IBMJMXConnectorREST. The port number is the HTTPS port.  
 You must also provide the user name and password.

- **8.5.5.4** Access the REST connector directly using an HTTPS call.

To use HTTPS calls to access REST connectors, you need WebSphere Application Server Liberty 8.5.5.4 or later.

- a. Open a browser at `https://<host>:<port>/IBMJMXConnectorREST/api`, and enter the administrative credentials you specified in step 3.
- b. Examine the available REST APIs. Each item has a description of its behavior, input, output, query parameters, and header.

**Note:** You can specify some JMX REST connection options as system properties. See the Liberty API - WebSphere JMX REST Connector API.

## Configuring web security related properties in Liberty

You can configure web security related properties for Liberty, such as SSO and client certificate authentication.

### About this task

You can use the `webAppSecurity` element to configure web container application security for Liberty. Make sure you add the `appSecurity-2.0`, `servlet-3.0` and other required Liberty features to the `server.xml` file.

For all available attributes in the `webAppSecurity` element, see **\*\*\*\* MISSING FILE \*\*\*\***.

You can choose to complete one or more of the following tasks according to your requirements.

### Procedure

- “Customizing SSO configuration using LTPA cookies in Liberty” on page 1203
- “Configuring your web application and server for client certificate authentication” on page 1164

## Customizing SSO configuration using LTPA cookies in Liberty

With single sign-on (SSO) configuration support, web users can authenticate once when accessing Liberty resources such as HTML, JavaServer Pages (JSP) files, and servlets, or accessing resources in multiple Liberty servers that share the same Lightweight Third Party Authentication (LTPA) keys.

### Example

When a user passes authentication on one of Liberty servers, authentication information generated by the server is transported to the web browser in a cookie. The cookie is used to propagate the authentication information to other Liberty servers.

The LTPA is configured and ready for immediate use. The default cookie name used to store the SSO token is called `ltpaToken2`. If you want to use a different name for the cookie, you can customize the cookie name using the `ssoCookieName` attribute of the `<webAppSecurity>` element. If you customize the cookie name, make sure that all the servers that participate in SSO use the same cookie name.

For more information about SSO, see SSO concept in Liberty.

The following example code sets the user to be logged out after the HTTP session expires and the name of the SSO cookie as myCookieName:

```
<webAppSecurity logoutOnHttpSessionExpire="true" ssoCookieName="myCookieName" />
```

**Note:** For SSO to work across Liberty servers, full profile servers, or both, set the following resources:

- The servers must use the same LTPA keys and share the same user registry.
- If the servers are not in the same domain, use the **ssoDomainNames** attribute of the <webAppSecurity> element to list the domains. The following example code sets the domain name to domain.com:

```
<webAppSecurity ssoDomainNames="domain.com" />
```

- If the servers are in the same domain, set the **ssoUseDomainFromURL** attribute of the <webAppSecurity> element to true, or specify the domain name in the **ssoDomainNames** attribute. The following example code sets **ssoUseDomainFromURL** to true so that the domain name is taken from the request URL:

```
<webAppSecurity ssoUseDomainFromURL="true" />
```

For details of all the available SSO settings, see the <webAppSecurity> element in \*\*\*\* MISSING FILE \*\*\*\*.

## Configuring your web application and server for client certificate authentication

You can configure your web application on Liberty using SSL client authentication.

### Before you begin

This topic assumes that you have already created the SSL certificates, for example as described in “Creating SSL certificates from the command line” on page 1161.

### About this task

Client certificate authentication occurs if the server-side requests that the client-side send a certificate. A WebSphere server can be configured for client certificate authentication on the SSL configuration. To do this, you add the ssl-1.0 Liberty feature to the server.xml file, along with code that tells the server the keystore information for authentication.

For details of which aspects of SSL are supported, see “Liberty features” on page 483.

### Procedure

1. Ensure that the deployment descriptor for your web application specifies client certificate authentication as the authentication method to use.

Check that the deployment descriptor includes the following element:

```
<auth-method>CLIENT-CERT</auth-method>
```

**Note:** You can use a tool such as Rational Application Developer to create the deployment descriptor.

2. Optional: Generate an SSL certificate using the command line. See “securityUtility command” on page 1162.
3. Configure your server to enable SSL client authentication by adding the following lines to the server.xml file:

```
<featureManager>
 <feature>ssl-1.0</feature>
</featureManager>

<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
```

```
trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
<keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="defaultPWD" />
<keyStore id="defaultTrustStore" location="trust.jks" type="JKS" password="defaultPWD" />
```

- If you specify `clientAuthentication="true"`, the server requests that a client sends a certificate. However, if the client does not have a certificate, or the certificate is not trusted by the server, the handshake does not succeed.
  - If you specify `clientAuthenticationSupported="true"`, the server requests that a client sends a certificate. However, if the client does not have a certificate, or the certificate is not trusted by the server, the handshake might still succeed.
  - If you do not specify either `clientAuthentication` or `clientAuthenticationSupported`, or you specify `clientAuthentication="false"` or `clientAuthenticationSupported="false"`, the server does not request that a client send a certificate during the handshake.
4. Add a client certificate to your browser. See the documentation of your browser for adding client certificates.
  5. Make sure the server trusts any client certificates that are used.
  6. Make sure any client certificates used for client authentication are mapped to a user identity in your registry.
    - For the basic registry, the user identity is the common name (CN) from the distinguished name (DN) of the certificate.
    - For a Lightweight Directory Access Protocol (LDAP) registry, the DN from the client certificate must be in the LDAP registry.
  7. To use basic authentication, user ID and password only, if client certificate authentication does not succeed, add the following line to your `server.xml` file.

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

**Note:** If you specify `allowFailOverToBasicAuth="false"` or do not specify `allowFailOverToBasicAuth`, and the client certificate authentication does not succeed, the request generates a 403 Authentication error message, and the client is not prompted for basic authentication.

## Configuring the Liberty server to track logged out LTPA tokens

8.5.5.4

You can configure a Liberty server to track logged out Lightweight Third Party Authentication (LTPA) tokens.

### About this task

When a user is logged out by using either form logout or programmatic logout, the LTPA token that is used for Single Sign On is removed from the cookie. The LTPA token that is used for SSO is also removed from the local Authentication cache and the session is invalidated. If the token was persisted and presented again, it is validated based on the expiration time and the LTPA encryption keys.

With this element enabled, the LTPA SSO tokens that were logged out on the server are tracked and if presented again on the same server are not used.. A logout is performed and the user needs to authenticate again.

This configuration only works on the same server. This means that the LTPA token can only be tracked on the server where the user logged out, and if that same LTPA token is presented to another server it is used and if the LTPA keys are shared and the token has not expired it is used until it is also logged out on that server.

## Procedure

To track the tokens that are logged out on a particular Liberty server, you can enable the following element in the `server.xml`:

```
<webAppSecurity trackLoggedOutSSOCookies="true"/>
```

When this element is enabled, it might affect your Single Sign On (SSO) scenarios. For example, if the user 'bob' logs in from multiple browsers to the same server and logs out from one browser and tries to access the resource by using another browser, the user must log in as the token presented is discarded.

---

## Configuring authentication aliases for Liberty

You can configure an authentication data alias to use with a resource reference for authentication in Liberty.

### About this task

You can use an authentication data alias by defining a user and password for authentication in Liberty. To do this, add the `jdbc-4.0` Liberty feature to the `server.xml` file and add at least one `authData` element.

**Note:** There is no authentication alias principal mapping module support.

### Procedure

1. Add the `jdbc-4.0` Liberty features in the `server.xml` file.

```
<featureManager>
 <feature>jdbc-4.0</feature>
</featureManager>
```

2. Configure the `authData` element in the `server.xml` file as follows. If the `authData` element is used as a top-level configuration element, you must set the `id` attribute value to a unique authentication alias.

```
<authData id="auth1" user="dbuser1" password="dbuser1pwd"/>
```

3. Configure the IBM deployment descriptor, for example, the `ibm-web-bnd.xml` file, of your application by using the `authentication-alias` element in the resource reference. The name attribute value must match the `id` attribute defined in the `server.xml` file.

```
<resource-ref name="jdbc/mydbresource" binding-name="jdbc/mydbresource">
 <authentication-alias name="auth1"/>
</resource-ref>
```

---

## Configuring JAAS for database authentication

8.5.5.9

You can use Java Authentication and Authorization Service (JAAS) for database authentication.

### About this task

You can use a JAAS login context entry to specify a custom login module to use for setting the user name and password to authenticate to a database.

### Procedure

1. Add the `appSecurity-2.0`, `jdbc-4.0`, and `jca-1.6` features in the `server.xml` file. You can also add `appSecurity-2.0`, `jdbc-4.1`, and `jca-1.7`. For example:

```

<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>jdbc-4.0</feature>
 <feature>jca-1.6</feature>
</featureManager>

```

2. Configure a `jaasLoginContextEntry` element in the `server.xml` file with the login module to use. For example:

```

<jaasLoginContextEntry id="myJAASLoginEntry" name="myJAASLoginEntry" loginModuleRef="myLoginModule" />
<jaasLoginModule id="myLoginModule" className="my.package.MyLoginModule" controlFlag="REQUIRED" libraryRef="customL

<library id="customLoginLib">
 <fileset dir="{server.config.dir}" includes="MyLoginModule.jar"/>
</library>

```

3. Configure the `dataSource` element's `jaasLoginContextEntry` attribute with the id of the `jaasLoginContextEntry` element configured in step 2. For example:

```

<dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2"
 jaasLoginContextEntry="myJAASLoginEntry" .../>

```

4. As an alternative to step 3, you can configure a `custom-login-configuration` element in the deployment descriptor `ibm-web-bnd.xml` file of your application. The name attribute must match the id attribute for `jaasLoginContextEntry` that is defined in the `server.xml` file. For example:

```

<resource-ref name="jdbc/ds1ref" binding-name="jdbc/ds1">
 <custom-login-configuration name="myJAASLoginEntry">
 <property name="property1" value="value1"/>
 </custom-login-configuration>
</resource-ref>

```

---

## Developing extensions to the Liberty security infrastructure

The Liberty server provides various plug-in points so that you can extend the security infrastructure.

### About this task

The following topics are covered in this section:

#### Procedure

- Follow the instructions in “Developing a custom TAI for Liberty” to develop custom trust association interceptors (TAI) to extend the security infrastructure of Liberty server.
- Follow the instructions in “Developing JAAS custom login modules for a system login configuration” on page 1305 to develop JAAS custom login modules to extend the security infrastructure of Liberty server.

## Developing a custom TAI for Liberty

You can develop a custom trust association interceptor (TAI) class by implementing the `com.ibm.wsspi.security.tai.TrustAssociationInterceptor` interface provided in the Liberty server.

### About this task

The trust association interface is a service provider API that enables the integration of third-party security services with a Liberty server. When processing the web request, the Liberty server calls out and passes the `HttpServletRequest` and `HttpServletResponse` to the trust association interceptors. The `HttpServletRequest` calls the `isTargetInterceptor` method of the interceptor to see whether the interceptor can process the request. After an appropriate trust association interceptor is selected, the `HttpServletRequest` is processed by the `negotiateValidateandEstablishTrust` method of the interceptor, and the result is returned in a `TAIResult` object. You can add your own logic code to each method of the custom TAI class.

See also the Java API document for the TAI interface. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `#{wlp.install.dir}/dev` directory.

**Avoid trouble:** **Distributed operating systems** There are several security configuration examples on the WASdev.net website for reference when configuring security for your applications on Liberty. See “Configuring TAI on Liberty by using developer tools” on page 1206.

## Example

Here is a sample TAI class called SimpleTAI, which also lists all available methods from the TrustAssociationInterceptor interface.

```
package com.ibm.websphere.security.sample;

import java.util.Properties;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.ibm.websphere.security.WebTrustAssociationException;
import com.ibm.websphere.security.WebTrustAssociationFailedException;
import com.ibm.wsspi.security.tai.TAIResult;
import com.ibm.wsspi.security.tai.TrustAssociationInterceptor;

public class SimpleTAI implements TrustAssociationInterceptor {
 public SimpleTAI() {
 super();
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#isTargetInterceptor
 * (javax.servlet.http.HttpServletRequest)
 */
 public boolean isTargetInterceptor(HttpServletRequest req)
 throws WebTrustAssociationException {
 //Add logic to determine whether to intercept this request
 return true;
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#negotiateValidateandEstablishTrust
 * (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)
 */
 public TAIResult negotiateValidateandEstablishTrust(HttpServletRequest req,
 HttpServletResponse resp) throws WebTrustAssociationFailedException {
 // Add logic to authenticate a request and return a TAI result.
 String tai_user = "taiUser";
 return TAIResult.create(HttpServletResponse.SC_OK, tai_user);
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#initialize(java.util.Properties)
 */
 public int initialize(Properties arg0)
 throws WebTrustAssociationFailedException {
 return 0;
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getVersion()
 */
}
```

```

 */
 public String getVersion() {
 return "1.0";
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getType()
 */
 public String getType() {
 return this.getClass().getName();
 }

 /*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#cleanup()
 */
 public void cleanup()

 {}
}

```

## What to do next

Add the TAI class to the Liberty server.

Use one of the following methods to add the TAI class to the Liberty server:

- Put the custom TAI class in a JAR file, for example `simpleTAI.jar`, then make the JAR file available as a shared library. See “Configuring TAI in Liberty” on page 1205.
- Package the custom TAI class as a feature. See “Developing a custom TAI as a Liberty feature” on page 1141.

## Developing a custom SIP TAI

8.5.5.7

When you develop Session Initiation Protocol (SIP) applications, you can create a custom trust association interceptor (TAI).

### Before you begin

Developing a SIP TAI is similar to developing any other custom interceptors used in trust associations. In fact, a custom TAI for a SIP application is an extension of the trust association interceptor model.

### About this task

TAI can be invoked by a SIP servlet request or a SIP servlet response. To implement a custom SIP TAI, you need to write your own Java class.

### Procedure

1. Write a Java class that extends the `com.ibm.wsspi.security.tai.extension.BaseTrustAssociationInterceptor` class and implements the `com.ibm.websphere.security.tai.extension.SIPTrustAssociationInterceptor` interface. Those classes are defined in the `${wlp.install.dir}/dev/api/ibm/ccom.ibm.websphere.appserver.api.sipServletSecurity.1.0_1.0.10.jar` file.
2. Declare the following Java methods:

```
public int initialize(Properties properties) throws WebTrustAssociationFailedException;
```

This is invoked before the first message is processed so that the implementation can allocate any resources that it needs. For example, it might establish a connection to a database.

WebTrustAssociationFailedException is defined in the `com.ibm.websphere.security_1.0.10.jar` file. The value of the properties argument comes from the `<trustAssociation>` configuration.

**public void cleanup();**

This is invoked when the TAI can free any resources that it holds. For example, it could close a connection to a database.

**public boolean isTargetProtocolInterceptor(SipServletMessage sipMsg) throws WebTrustAssociationFailedException;**

Your custom TAI can use this method to handle the sipMsg message. If the method returns false, WebSphere ignores your TAI for sipMsg.

**public TAIResult negotiateValidateandEstablishProtocolTrust (SipServletRequest req, SipServletResponse resp) throws WebTrustAssociationFailedException;**

This method returns a TAIResult that indicates the status of the message that is being processed and a user ID or the unique ID for the user who is trying to authenticate. If authentication succeeds, the TAIResult contains the status `HttpServletResponse.SC_OK` and a principal. If authentication fails, the TAIResult will contain a return code of `HttpServletResponse.SC_UNAUTHORIZED (401)`, `SC_FORBIDDEN (403)`, or `SC_PROXY_AUTHENTICATION_REQUIRED (407)`. This only indicates whether the container should accept a message for further processing. To challenge an incoming request, the TAI implementation must generate and send its own `SipServletResponse` containing a challenge. The exception can be thrown for internal TAI errors. Table 1 describes the argument values and resultant actions for the `negotiateValidateandEstablishProtocolTrust` method.

Table 92. Description of `negotiateValidateandEstablishProtocolTrust` arguments and actions.

This table provides a description of the `negotiateValidateandEstablishProtocolTrust` arguments and actions

Argument or action	For a SIP request	For a SIP response
Value of req argument	The incoming request	Null
Value of resp argument	Null	The incoming response
Action for valid response credentials	Return <code>TAIResult.status</code> containing <code>SC_OK</code> and a user ID or unique ID	Return <code>TAIResult.status</code> containing <code>SC_OK</code> and a user ID or unique ID
Action for incorrect response credentials	Return the <code>TAIResult</code> with the 4xx status	Return the <code>TAIResult</code> with the 4xx status

The sequence of events is as follows:

- a. The SIP container maps initial requests to applications by using the rules in each applications deployment descriptor; subsequent messages are mapped based on JSR289 mechanisms.
- b. If any of the applications require security, the SIP container invokes any defined TAI implementations for the message.
- c. If the message passes security, the container invokes the corresponding applications.

Your TAI implementation can modify a SIP message, but the modified message will not be usable within the request mapping process, because it finishes before the container invokes the TAI.

The `com.ibm.wsspi.security.tai.TAIResult` class, which is defined in the `com.ibm.ws.security.authentication.tai_1.0.10.jar` file, has three static methods for creating a `TAIResult`. The `TAIResult` create methods take an int type as the first parameter. The WebSphere Application Server expects the result to be a valid HTTP request return code and is interpreted as follows:



If the value is `HttpServletResponse.SC_OK`, this response tells WebSphere that the TAI has completed its negotiation. The response also tells WebSphere to use the information in the `TAIResult` to create a user identity.

The created `TAIResults` have the meanings that are shown in Table 2.

Table 93. Meanings of `TAIResults`.

This table lists the meanings of `TAIResults`

TAIResult	Explanation
<code>public static TAIResult create(int status);</code>	Indicates a status to the WebSphere Application Server. The status should not be <code>SC_OK</code> because the identity information is provided.
<code>public static TAIResult create(int status, String principal);</code>	Indicates a status to the WebSphere Application Server and provides the user ID or the unique ID for this user. WebSphere creates credentials by querying the user registry.
<code>public static TAIResult create(int status, String principal, Subject subject);</code>	Indicates a status to the WebSphere Application Server, the user ID or the unique ID for the user, and a custom Subject. If the Subject contains a Hashtable, the principal is ignored. The contents of the Subject becomes part of the eventual user Subject.

**public String getVersion();**

This method returns the version number of the current TAI implementation.

**public String getType();**

This method's return value is implementation-dependent.

3. Compile the implementation after you have implemented it to create your own SIP TAI jar file.
4. Follow steps 3-4 described in the topic *Configuring TAI for Liberty* to configure the Liberty server to use the SIP TAI.

## Developing JAAS custom login modules for a system login configuration

For a Liberty server, multiple Java Authentication and Authorization Service (JAAS) plug-in points exist for configuring system logins. The Liberty uses system login configurations to authenticate incoming requests. You can develop a custom JAAS login module to add information to the **Subject** of a system login configuration.

### About this task

Application login configurations are called by servlet applications for obtaining a Subject that is based on specific authentication information. When you write a login module that plugs into a Liberty profile application login or system login configuration, you must develop login configuration logic that knows when specific information is present, and how to use the information. See “JAAS configuration” on page 587 and “JAAS login modules” on page 588 for more details.

To develop a JAAS custom login module for a system login configuration, follow the steps in the procedure:

### Procedure

- Understand usable callbacks and how they work.  
See *Programmatic login for JAAS* for more information about usable callbacks.

**Note:** Liberty only supports the following callbacks:

```
callbacks[0] = new javax.security.auth.callback.NameCallback("Username: ");
callbacks[1] = new javax.security.auth.callback.PasswordCallback("Password: ", false);
callbacks[2] = new com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl("Credential Token: ");
```

```

callbacks[3] = new com.ibm.websphere.security.auth.callback.WSServletRequestCallback("HttpServletRequest: ");
callbacks[4] = new com.ibm.websphere.security.auth.callback.WSServletResponseCallback("HttpServletResponse: ");
callbacks[5] = new com.ibm.websphere.security.auth.callback.WSAppContextCallback("ApplicationContextCallback: ");
callbacks[6] = new WSRealmNameCallbackImpl("Realm Name: ", default_realm);
callbacks[7] = new WSX509CertificateChainCallback("X509Certificate[]: ");
callbacks[8] = wsAuthMechOidCallback = new WSAuthMechOidCallbackImpl("AuthMechOid: ");

```

- Understand shared state variables and how they work.

If you want to access the objects that the WebSphere Application Server traditional creates during a login, refer to the following shared state variables. For more information about these variables, see the “System Programming Interfaces” subtopic of Programming Interfaces.

#### **com.ibm.wsspi.security.auth.callback.Constants.WSPRINCIPAL\_KEY**

Specifies an implemented object of the java.security.Principal interface. This shared state variable is for read-only purposes. Do not set this variable in the shared state for custom login modules. The default login module sets this variable.

#### **com.ibm.wsspi.security.auth.callback.Constants.WSCREDENTIAL\_KEY**

Specifies the com.ibm.websphere.security.cred.WSCredential object. This shared state variable is for read-only purposes. Do not set this variable in the shared state for custom login modules. The default login module will set this variable.

#### **com.ibm.wsspi.security.auth.callback.Constants.WSSSOTOKEN\_KEY**

Specifies the com.ibm.wsspi.security.token.SingleSignonToken object. Do not set this variable in the shared state for custom login modules. The default login module sets this variable.

- Optional: Understand hashtables for custom JAAS login modules in Liberty. See “Hash table login module” on page 594 for more details.
- Develop a sample custom login module using callbacks and shared state.

You can use the following sample to learn on how to use some of the callbacks and shared state variables.

```

public class CustomCallbackLoginModule implements LoginModule {

 protected Map<String, ?> _sharedState;
 protected Subject _subject = null;
 protected CallbackHandler _callbackHandler;
 private final String customPrivateCredential = "CustomLoginModuleCredential";

 /**
 * Initialization of login module
 */
 public void initialize(Subject subject, CallbackHandler callbackHandler,
 Map<String, ?> sharedState, Map<String, ?> options) {
 _sharedState = sharedState;
 _subject = subject;
 _callbackHandler = callbackHandler;
 }

 public boolean login() throws LoginException {
 try {
 AccessController.doPrivileged(new PrivilegedExceptionAction<Object>() {
 public Object run() throws Exception {
 _subject.getPrivateCredentials().add(customPrivateCredential);
 return null;
 }
 });
 } catch (PrivilegedActionException e) {
 throw new LoginException(e.getLocalizedMessage());
 }

 String username = null;
 char passwordChar[] = null;
 byte[] credToken = null;
 HttpServletRequest request = null;
 HttpServletResponse response = null;
 Map appContext = null;
 String realm = null;
 String authMechOid = null;
 java.security.cert.X509Certificate[] certChain = null;

```

```

NameCallback nameCallback = null;
PasswordCallback passwordCallback = null;
WSCredTokenCallbackImpl wsCredTokenCallback = null;
WSServletRequestCallback wsServletRequestCallback = null;
WSServletResponseCallback wsServletResponseCallback = null;
WSAppContextCallback wsAppContextCallback = null;
WSRealmNameCallbackImpl wsRealmNameCallback = null;
WSX509CertificateChainCallback wsX509CertificateCallback = null;
WSAuthMechOidCallbackImpl wsAuthMechOidCallback = null;

Callback[] callbacks = new Callback[9];
callbacks[0] = nameCallback = new NameCallback("Username: ");
callbacks[1] = passwordCallback = new PasswordCallback("Password: ", false);
callbacks[2] = wsCredTokenCallback = new WSCredTokenCallbackImpl("Credential Token: ");
callbacks[3] = wsServletRequestCallback = new WSServletRequestCallback("HttpServletRequest: ");
callbacks[4] = wsServletResponseCallback = new WSServletResponseCallback("HttpServletResponse: ");
callbacks[5] = wsAppContextCallback = new WSAppContextCallback("ApplicationContextCallback: ");
callbacks[6] = wsRealmNameCallback = new WSRealmNameCallbackImpl("Realm name:");
callbacks[7] = wsX509CertificateCallback = new WSX509CertificateChainCallback("X509Certificate[]: ");
callbacks[8] = wsAuthMechOidCallback = new WSAuthMechOidCallbackImpl("AuthMechOid: ");

try {
 _callbackHandler.handle(callbacks);
} catch (Exception e) {
 // handle exception
}

if (nameCallback != null)
 username = nameCallback.getName();

if (passwordCallback != null)
 passwordChar = passwordCallback.getPassword();

if (wsCredTokenCallback != null)
 credToken = wsCredTokenCallback.getCredToken();

if (wsServletRequestCallback != null)
 request = wsServletRequestCallback.getHttpServletRequest();

if (wsServletResponseCallback != null)
 response = wsServletResponseCallback.getHttpServletResponse();

if (wsAppContextCallback != null)
 appContext = wsAppContextCallback.getContext();

if (wsRealmNameCallback != null)
 realm = wsRealmNameCallback.getRealmName();

if (wsX509CertificateCallback != null)
 certChain = wsX509CertificateCallback.getX509CertificateChain();

if (wsAuthMechOidCallback != null)
 authMechOid = wsAuthMechOidCallback.getAuthMechOid();

_subject.getPrivateCredentials().add("username = " + username);
_subject.getPrivateCredentials().add("password = " + String.valueOf(passwordChar));
_subject.getPrivateCredentials().add("realm = " + realm);
_subject.getPrivateCredentials().add("authMechOid = " + authMechOid.toString());

return true;
}

public boolean commit() throws LoginException {
 return true;
}

public boolean abort() {
 return true;
}

public boolean logout() {
 return true;
}
}

```

- Optional: Develop a sample custom login module using hashtable login. You can use the following sample to learn on how to use hashtable login.

```

package com.ibm.websphere.security.sample;

import java.util.Map;

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginException;
import javax.security.auth.spi.LoginModule;

import com.ibm.wsspi.security.token.AttributeNameConstants;

/**
 * Custom login module that adds another PublicCredential to the subject
 */
@SuppressWarnings("unchecked")
public class CustomHashtableLoginModule implements LoginModule {

 protected Map<String, ?> _sharedState;
 protected Map<String, ?> _options;

 /**
 * Initialization of login module
 */
 public void initialize(
 Subject subject, CallbackHandler callbackHandler, Map<String, ?> sharedState, Map<String, ?> options) {
 _sharedState = sharedState;
 _options = options;
 }

 public boolean login() throws LoginException {
 try {
 java.util.Hashtable<String, Object> customProperties = (java.util.Hashtable<String, Object>)
 _sharedState.get(AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY);
 if (customProperties == null) {
 customProperties = new java.util.Hashtable<String, Object>();
 }

 customProperties.put(AttributeNameConstants.WSCREDENTIAL_USERID, "userId");
 // Sample of creating custom cache key
 customProperties.put(AttributeNameConstants.WSCREDENTIAL_CACHE_KEY, "customCacheKey");

 /*
 * Sample for creating user ID and security name
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_UNIQUEID, "userId");
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_SECURITYNAME, "securityName");
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_REALM, "realm");
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_GROUPS, "groupList");
 */
 /*
 * Sample for creating user ID and password
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_USERID, "userId");
 * customProperties.put(AttributeNameConstants.WSCREDENTIAL_PASSWORD, "password");
 */
 Map<String, java.util.Hashtable> mySharedState = (Map<String, java.util.Hashtable>) _sharedState;
 mySharedState.put(AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY, customProperties);
 } catch (Exception e) {
 throw new LoginException("LoginException: " + e.getMessage());
 }

 return true;
 }

 public boolean commit() throws LoginException {
 return true;
 }

 public boolean abort() {
 return true;
 }

 public boolean logout() {
 return true;
 }
}

```

## What to do next

Add your custom login module into the WEB\_INBOUND, and DEFAULT Java Authentication and Authorization Service (JAAS) system login configurations of the server.xml file. Put the custom login module class in a JAR file, for example, customLoginModule.jar, then make the JAR file available to the Liberty server. See “Configuring a JAAS custom login module for Liberty” on page 1184.

## Developing a custom JASPIC authentication provider for Liberty



8.5.5.6

You can develop a custom Java Authentication SPI for Containers (JASPIC) authentication provider by creating classes that implement the required interfaces noted in the JSR 196: Java Authentication Service Provider Interface for Containers specification.

### Before you begin

Review the specific interface implementation requirements for JASPIC authentication providers and modules in the JSR 196: Java Authentication Service Provider Interface for Containers specification.

### About this task

WebSphere Application Server Liberty supports the use of third-party authentication providers that are compliant with the servlet container profile specified in Java Authentication SPI for Containers (JASPIC) Version 1.1.

The servlet container profile defines interfaces that are used by the security runtime environment in collaboration with the web container in the WebSphere Application Server to invoke authentication modules before and after a web request is processed by an application. Authentication that uses JASPIC modules is performed only when JASPIC is enabled in the security configuration.

To develop a custom authentication provider, create classes that implement the required interfaces noted in the JSR 196: Java Authentication Service Provider Interface for Containers specification. A provider can use one or more authentication modules for authentication. Modules can use callbacks to perform authentication, or they can manually add the necessary user identity information to the client subject.

### Procedure

1. Create a class that implements the `javax.security.auth.message.config.AuthConfigProvider` interface.

The `AuthConfigProvider` implementation class must define a public two-argument constructor and the `getServerAuthConfig` public method:

```
import java.util.Map;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.config.AuthConfigFactory;
import javax.security.auth.message.config.AuthConfigProvider;
import javax.security.auth.message.config.ServerAuthConfig;

public class SampleAuthConfigProvider implements AuthConfigProvider {

 public SampleAuthConfigProvider(Map<String, String> properties, AuthConfigFactory factory) {
 ...
 }
 public ServerAuthConfig getServerAuthConfig(String layer, String appContext, CallbackHandler handler)
 throws AuthException {
 ...
 }
}
```

An instance of the `AuthConfigProvider` implementation class is used by WebSphere Application Server when a request arrives to be processed by the web module of the application. The `getServerAuthConfig` method is used to obtain a `ServerAuthConfig` instance. The `CallbackHandler` argument in the method call is used by the authentication module.

2. Create a class that implements the `javax.security.auth.message.config.ServerAuthConfig` interface. The `ServerAuthConfig` implementation class must define the `getAuthContextID` and `getAuthContext` public methods:

```
import java.util.Map;
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.config.ServerAuthConfig;
import javax.security.auth.message.config.ServerAuthContext;

public class SampleServerAuthConfig implements ServerAuthConfig {

 public String getAuthContextID(MessageInfo messageInfo) throws IllegalArgumentException {
 ...
 }
 public ServerAuthContext getAuthContext(String authContextID, Subject serviceSubject, Map properties)
 throws AuthException {
 ...
 }
}
```

The `getAuthContextID` and `getAuthContext` methods in the `ServerAuthConfig` implementation class are used to obtain a `ServerAuthContext` instance.

3. Create a class that implements the `javax.security.auth.message.config.ServerAuthContext` interface. The `ServerAuthContext` implementation class must define the `validateRequest` and `secureResponse` public methods:

```
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.config.ServerAuthContext;

public class SampleServerAuthContext implements ServerAuthContext {

 public AuthStatus validateRequest(MessageInfo messageInfo, Subject clientSubject, Subject serviceSubject)
 throws AuthException {
 ...
 }
 public AuthStatus secureResponse(MessageInfo messageInfo, Subject serviceSubject)
 throws AuthException {
 ...
 }
}
```

The `validateRequest` method in the `ServerAuthContext` implementation class is used to invoke the module that authenticates the received web request message. If the authentication result is successful, the web container dispatches the received web request message that the target web module processes in the application. If the authentication result is not successful, the request is rejected with the appropriate response status.

4. Create a class that implements the `javax.security.auth.message.module.ServerAuthModule` interface. The `ServerAuthModule` implementation class must define the `initialize`, `validateRequest`, and `secureResponse` public methods:

```
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.MessagePolicy;
```

```

import javax.security.auth.message.module.ServerAuthModule;

public class SampleAuthModule implements ServerAuthModule {

 public void initialize(MessagePolicy requestPolicy, MessagePolicy responsePolicy, CallbackHandler handler, MessageInfo messageInfo)
 throws AuthException {
 ...
 }

 public AuthStatus validateRequest(MessageInfo messageInfo, Subject clientSubject, Subject serviceSubject)
 throws AuthException {
 ...
 }

 public AuthStatus secureResponse(MessageInfo messageInfo, Subject serviceSubject)
 throws AuthException {
 ...
 }

 public void cleanSubject(MessageInfo messageInfo, Subject subject)
 throws AuthException {
 ...
 }
}

```

The `initialize` method in the `ServerAuthModule` implementation class is called by the `ServerAuthContext` implementation class to initialize the authentication module and to associate it with the `ServerAuthContext` instance.

The `validateRequest` and `secureResponse` methods in this class are used to authenticate the `javax.servlet.http.HttpServletRequest` and `javax.servlet.http.HttpServletResponse` contained in the `javax.security.auth.message.MessageInfo` that is received. These methods can use the `CallbackHandler` instance that is received in the `initialize` method to interact with the WebSphere security run time to validate a user password, and the active user registry to retrieve a unique id and group membership for a user. The retrieved data is placed in a `Hashtable` in the set of private credentials in the client subject. The WebSphere Application Server implementation of the `CallbackHandler` supports the following three callbacks:

- `CallerPrincipalCallback`
- `GroupPrincipalCallback`
- `PasswordValidationCallback`

WebSphere Application Server expects the name values obtained with `PasswordValidationCallback.getUsername()` and `CallerPrincipalCallback.getName()` to be identical. If they are not, unpredictable results occur. The `handle()` method of the `CallbackHandler` processes each callback that is given in the argument array of the method sequentially. Therefore, the name value set in the private credentials of the client subject is the one obtained from the last callback processed.

If `CallbackHandler` is not used by the authentication module, and `validateRequest` returns a successful status, WebSphere Application Server requires that a `Hashtable` instance be included in the `clientSubject` with user identity information so that a custom login can be performed to obtain the credentials for the user. This `Hashtable` can be added to the client subject as in the following example:

```

import java.util.Hashtable;
import java.util.String;
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import com.ibm.wsspi.security.registry.RegistryHelper;
import com.ibm.wsspi.security.token.AttributeNameConstants.AttributeNameConstants;

public AuthStatus validateRequest(MessageInfo messageInfo, Subject clientSubject, Subject serviceSubject)
 throws AuthException {
 ...
}

```

```

UserRegistry reg = RegistryHelper.getUserRegistry(null);
String uniqueid = reg.getUniqueUserID(username);

Hashtable hashtable = new Hashtable();
hashtable.put(AttributeConstants.WSCREDENTIAL_UNIQUEID, uniqueid);
hashtable.put(AttributeConstants.WSCREDENTIAL_SECURITYNAME, username);
hashtable.put(AttributeConstants.WSCREDENTIAL_PASSWORD, password);
hashtable.put(AttributeConstants.WSCREDENTIAL_GROUPS, groupList); //optional
clientSubject.getPrivateCredentials().add(hashtable);
...
}

```

For more information about the Hashtable requirements and custom login, see Developing JAAS custom login modules for a system login configuration.

## Developing a Java Authorization Contract for Containers (JACC) Authorization Provider



8.5.5.6

You can develop a JACC provider to have custom authorization decisions for Java Platform, Enterprise Edition (J2EE) applications by implementing the `com.ibm.wsspi.security.authorization.jacc.ProviderService` interface that is provided in the Liberty server.

### Before you begin

By default, the application module loading is deferred until the request to the application is being processed, however the security constraint of the entire module in the application needs to be processed before the application is ready to be processed. The deferred module loading needs to be disabled. The following shows you how to disable it:

1. For the WebContainer:

In the `server.xml` file, the following element needs to be set:

```
<webContainer deferServletLoad="false"/>
```

2. For the EJBContainer:

In the `server.xml` file, the following element needs to be set:

```
<ejbContainer startEJBsAtAppStart="true"/>
```

**Note:** If the previous elements are not set, the complete security constraint information may not be propagated to the third party JACC provider upon starting the server. As a result, the correct authorization decision may not be enforced by the third party JACC provider.

### About this task

The Java Authorization Contract for Containers specification, JSR 115, defines an interface for authorization providers. In the Liberty server, you must package your JACC provider as a user feature. Your feature must implement the `com.ibm.wsspi.security.authorization.jacc.ProviderService` interface.

### Procedure

1. Create an OSGi component that provides a service that implements the `com.ibm.wsspi.security.authorization.jacc.ProviderService` interface.

The `ProviderService` interface defines two methods, `getPolicy`, which the Liberty server run time invokes to retrieve an instance of your `Policy` class that implements the `java.security.Policy` abstract class, and `getPolicyConfigFactory`, which the Liberty server run time invokes to retrieve an instance



of your `PolicyConfigurationFactory` class that implements the `javax.security.jacc.PolicyConfigurationFactory` abstract class. The following example uses OSGi declarative services annotations:

```
package com.mycompany.jacc;

import com.mycompany.jacc.MyAuthConfigProvider;
import com.ibm.wsspi.security.authorization.jacc.ProviderService;
import java.security.Policy;
import java.util.Map;
import javax.security.jacc.PolicyConfigurationFactory;
import org.osgi.service.component.ComponentContext;
import org.osgi.service.component.annotations.Activate;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Deactivate;

// The property value of javax.security.jacc.policy.provider which defines the implementation class of Policy and
// javax.security.jacc.PolicyConfigurationFactory.provider which defines the implementation class of PolicyConfigurat

@Component(service = ProviderService.class,
 immediate = true,
 property = {
 "javax.security.jacc.policy.provider=com.myco.jacc.MyPolicy",
 "javax.security.jacc.PolicyConfigurationFactory.provider="
 + "com.myco.jacc.MyFactoryImpl"
 }
)

public class MyJaccProviderService implements ProviderService {
 Map<String, String> configProps;

 // This method called by the Liberty runtime
 // to get an instance of Policy class
 @Override
 public Policy getPolicy() {
 return new myPolicy();
 }

 // This method called by the Liberty runtime
 // to get an instance of PolicyConfigurationFactory class
 @Override
 public PolicyConfigurationFactory getPolicyConfigurationFactory() {
 ClassLoader cl = null;
 PolicyConfigurationFactory pcf = null;
 System.setProperty(
 "javax.security.jacc.PolicyConfigurationFactory.provider",
 "com.myco.jacc.MyFactoryImpl");
 try {
 cl = Thread.currentThread().getContextClassLoader();
 Thread.currentThread().setContextClassLoader(
 this.getClass().getClassLoader());
 pcf = PolicyConfigurationFactory.getPolicyConfigurationFactory();
 } catch (Exception e) {
 return null;
 } finally {
 Thread.currentThread().setContextClassLoader(cl);
 }
 return pcf;
 }

 @Activate
 protected void activate(ComponentContext cc) {
 // Read provider config properties here if needed,
 // then pass them to the Provider ctor.
 // This example reads the properties from the OSGi
 // component definition.
 configProps = (Map<String, String>) cc.getProperties();
 }
}
```

```

 }

 @Deactivate
 protected void deactivate(ComponentContext cc) {}
}

```

2. Package the component into an OSGi bundle that is part of your user feature, along with your JACC provider.
3. Ensure that your feature includes the OSGi subsystem content: `com.ibm.ws.javaee.jacc.1.5; version="[1,1.0.100)"; location="dev/api/spec/"`.
4. After the feature is installed into the user product extension location, configure the `server.xml` file with the feature name. For example:

```

<featureManager>
 ...
 <feature>usr:myJaccProvider</feature>
</featureManager>

```

## Developing a customPasswordEncryption Provider

8.5.5.9

You can develop a `customPasswordEncryption` provider to have custom authorization decisions for Java Platform, Enterprise Edition (J2EE) applications by implementing the `com.ibm.wsspi.security.crypto.CustomPasswordEncryption` interface that is provided in the Liberty server.

### About this task

#### Procedure

1. Create an OSGi component that provides a service that implements the `com.ibm.wsspi.security.crypto.CustomPasswordEncryption` interface.

The `CustomPasswordEncryption` interface defines three methods, `decrypt`, which the Liberty server run time invokes to decrypt the string, `encrypt`, which the Liberty server run time invokes to encrypt the string, and `initialize`, which is reserved for future use.

The following example uses OSGi declarative services annotations:

```

package com.mycompany.custom;

import org.osgi.service.component.ComponentContext;
import org.osgi.service.component.annotations.Activate;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.ConfigurationPolicy;
import org.osgi.service.component.annotations.Deactivate;
import org.osgi.service.component.annotations.Modified;

import com.ibm.wsspi.security.crypto.CustomPasswordEncryption;
import com.ibm.wsspi.security.crypto.EncryptedInfo;
import com.ibm.wsspi.security.crypto.PasswordDecryptException;
import com.ibm.wsspi.security.crypto.PasswordEncryptException;

/**
 */
@Component(service = CustomPasswordEncryption.class,
 immediate = true,
 name = "com.mycompany.CustomPasswordEncryptionImpl",
 configurationPolicy = ConfigurationPolicy.OPTIONAL,
 property = { "someKey=someValue" })
public class CustomPasswordEncryptionImpl implements CustomPasswordEncryption {

 @Activate
 protected synchronized void activate(ComponentContext cc, Map<String, Object> props) {
 }
}

```

```

@Modified
protected synchronized void modify(Map<String, Object> props) {
}

@Deactivate
protected void deactivate(ComponentContext cc) {
}

/**
 * The encrypt operation takes a UTF-8 encoded String in the form of a byte[].
 * The byte[] is generated from String.getBytes("UTF-8"). An encrypted byte[]
 * is returned from the implementation in the EncryptedInfo object.
 * Additionally, a logically key alias is returned in EncryptedInfo so which
 * is passed back into the decrypt method to determine which key was used to
 * encrypt this password. The WebSphere Application Server runtime has no
 * knowledge of the algorithm or key used to encrypt the data.
 *
 * @param decrypted_bytes
 * @return com.ibm.wsspi.security.crypto.EncryptedInfo
 * @throws com.ibm.wsspi.security.crypto.PasswordEncryptException
 */
@Override
public EncryptedInfo encrypt(byte[] input) throws PasswordEncryptException {
 byte[] output = null;
 String key = null;
 try {
 :
 <do some encryption>
 :
 return new EncryptedInfo(output, key);
 } catch (Exception e) {
 throw new PasswordEncryptException("Exception is caught", e);
 }
}

/**
 * The decrypt operation takes the EncryptedInfo object containing a byte[]
 * and the logical key alias and converts it to the decrypted byte[]. The
 * WebSphere Application Server runtime will convert the byte[] to a String
 * using new String (byte[], "UTF-8");
 *
 * @param info
 * @return byte[]
 * @throws PasswordEncryptException
 * @throws com.ibm.wsspi.security.crypto.PasswordDecryptException
 */
@Override
public byte[] decrypt(EncryptedInfo info) throws PasswordDecryptException {
 byte[] input = info.getEncryptedBytes();
 String key = info.getKeyAlias();
 byte[] output = null;
 try {
 :
 <do some decryption>
 :
 return output;
 } catch (Exception e) {
 throw new PasswordEncryptException("Exception is caught", e);
 }
}

/**
 * This is reserved for future use and is currently not called by the
 * WebSphere Application Server runtime.
 *
 * @param initialization_data

```

```

 **/
 @SuppressWarnings("rawtypes")
 @Override
 public void initialize(Map initialization_data) {}
}

```

2. Package the component into an OSGi bundle that is part of your user feature. Make sure that the bundle includes the OSGi service manifest.

The following example shows the contents of OSGi service manifest:

```

<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="com.mycompany.custom.CustomPasswordEncryptionImpl">
 <implementation class="com.mycompany.custom.CusomPasswordEncryptionImpl"/>
 <service>
 <provide interface="com.ibm.wsspi.security.crypto.CustomPasswordEncryption"/>
 </service>
 <property name="<someKey>" type="String" value="<someValue>"/>
</scr:component>

```

3. Ensure that your feature manifest includes the OSGi subsystem content with start-phase="SERVICE\_EARLY". For example:

```

Manifest-Version: 1.0
IBM-Feature-Version: 2
IBM-ShortName: customPasswordEncryption-1.0
Subsystem-Type: osgi.subsystem.feature
Subsystem-Version: 1.0.0
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: customPasswordEncryption-1.0;visibility:=public
Subsystem-Content:
com.mycompany.custom; version="[1,1.0.100)"; start-phase="SERVICE_EARLY"

```

4. After the feature is installed into the user product extension location, configure the server.xml file with the feature name.

```

<featureManager>
 ...
 <feature>usr:customPasswordEncryption-1.0</feature>
</featureManager>

```

## Customizing an application login to perform an identity assertion by using JAAS

You can use the Java Authentication and Authorization Service (JAAS) login framework to create a JAAS login configuration that can be used to perform login to an identity assertion on Liberty.

### About this task

By configuring identity assertion with trust validation, an application can use the JAAS login configuration to perform a programmatic identity assertion. See `IdentityAssertionLoginModule` for more detail.

**Avoid trouble:** **Distributed operating systems** There are several security configuration examples on the [WASdev.net](http://WASdev.net) website for reference when configuring security for your applications on Liberty. See “Configuring JAAS on Liberty by using developer tools” on page 1186.

### Procedure

1. Delegate trust validation to a user-implemented plug-in point.

Trust validation is accomplished by a custom login module. This custom login module performs any trust validation required, then sets the trust and identity information in the shared state to be passed on to the identity assertion login module. A map is required in the following shared state key:

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

If the state is missing then a `WSLoginFailedException` problem is reported by the `IdentityAssertionLoginModule` class.

The map in the shared state key must include a trust key with the following key name:  
`com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trust`

If this key is set to `true`, then trust is established. If the key is set to `false`, then no trust is established and `IdentityAssertionLoginModule` class creates a `WSLoginFailedException` problem.

The map in the shared state key must also set one of the following resources:

- An identity key. A `java.security.Principal` can be set in the following key:  
`com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal`
- A `java.security.cert.X509Certificate[]`. This certificate can be set in the following key:  
`com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates`

If both a principal and certificate are supplied, then the principal is used and a warning is reported.

2. Create a JAAS configuration for application logins. The JAAS configuration will contain the user-implemented trust validation custom login module and `IdentityAssertionLoginModule` class. Then to configure an application login configuration, add the following code in the `server.xml` file:

```
<jaasLoginContextEntry id="CustomIdentityAssertion" name="CustomIdentityAssertion"
 loginModuleRef="customIdentityAssertion,identityAssertion" />
<jaasLoginModule id="customIdentityAssertion"
 className="com.ibm.ws.security.authentication.IdentityAssertionLoginModule"
 controlFlag="REQUIRED" libraryRef="customLoginLib"/>
<library id="customLoginLib">
 <fileset dir="{server.config.dir}" includes="IdentityAssertionLoginModule.jar"/>
</library>
```

This JAAS configuration is used by the application to perform an identity assertion.

3. Perform the programmable identity assertion. A program can now use the JAAS login configuration to perform a programmatic identity assertion. The application program can create a login context for the JAAS configuration created in step 2, then log in to that login context with the identity that would assert to. If the login is successful then that identity can be set in the current running process. The following example illustrates this process:

```
NameCallback handler = new NameCallback(new MyPrincipal("Joe"));
LoginContext lc = new LoginContext("customIdentityAssertion", handler);
lc.login(); //assume successful
Subject s = lc.getSubject();
WSSubject.setRunAsSubject(s);
// From here on , the runas identity is "Joe"
```

**Note:** The `MyPrincipal` class is the implementation of the `java.security.Principal` interface in the example.

## Results

Using the JAAS login framework and two user-implemented login modules, you can create a JAAS login configuration that can be used to log in to an identity assertion.

## Developing a custom user registry in Liberty

You can develop a custom user registry class by implementing the `com.ibm.websphere.security.UserRegistry` interface provided in the Liberty server.

## About this task

The UserRegistry interface is a Service Programming Interface (SPI) that enables support to virtually any type of account repository. For a general view of stand-alone custom registries, see Stand-alone custom registries.

### Procedure

1. Implement the custom user registry. For more information, see Developing the UserRegistry interface for using custom registries.
2. Convert the implementation class into an OSGi service. You can do the conversion in the following ways:
  - Convert your UserRegistry class into a Declarative Service (DS) component. For more information, see “Declaring your services to OSGi Declarative Services” on page 1116.
  - Write a new UserRegistry class that is a DS component and delegate it to your UserRegistry class.
  - Register your UserRegistry class directly in the Service Registry (SR) using the OSGi core APIs. For more information, see “Working with the OSGi service registry” on page 1112.
3. Package the custom user registry as an OSGi bundle and export the UserRegistry service. For information on creating an OSGi bundle, see Creating an OSGi service bundle.
4. Create a feature manifest to include the OSGi bundle. For more information, see “Product extension” on page 577.
5. After the feature is installed into the user product extension location, configure the server.xml file with the feature name. For example:

```
<featureManager>
 ...
 <feature>usr:customRegistrySample-1.0</feature>
</featureManager>
```

For a downloadable custom user registry sample, see [https://developer.ibm.com/wasdev/downloads/#asset/samples-Custom\\_User\\_Registry](https://developer.ibm.com/wasdev/downloads/#asset/samples-Custom_User_Registry).

For more information, see <https://www.ibm.com/developer/worksheets/samples/creating-a-custom-user-registry-as-a-liberty-user-feature/>.

## Developing JAAS custom login modules for database authentication

8.5.5.9

You can develop a Java Authentication and Authorization Service (JAAS) custom login module for adding a user name and password to authenticate to a database.

### About this task

You can develop a JAAS custom login module that can be invoked when a database connection that requires authentication is created. The JAAS custom login module is responsible of creating a password credential that contains the user name, password, and managed connection factory. The login module must add the password credential to the subject's private credentials set to be used to authenticate to the database.

### Procedure

1. Create a class that implements the `javax.security.auth.spi.LoginModule` interface.
2. Save the necessary fields in the initialize method. For example:

```
/** {@inheritDoc} */
@SuppressWarnings("unchecked")
@Override
public void initialize(Subject subject, CallbackHandler callbackHandler, Map<String, ?> sharedState, Map<String, ?> options) throws LoginException {
 this.callbackHandler = callbackHandler;
}
```

```

 this.subject = subject;
 this.sharedState = (Map<String, Object>) sharedState;
 this.options = options;
}

```

3. Handle the `WSManagedConnectionFactoryCallback` and `WSMappingPropertiesCallback` callbacks in the login method. For example:

```

/** {@inheritDoc} */
@Override
public boolean login() throws LoginException {
 ...
 Callback callbacks[] = new Callback[2];
 callbacks[0] = new WSManagedConnectionFactoryCallback("Target ManagedConnectionFactory: ");
 callbacks[1] = new WSMappingPropertiesCallback("Mapping Properties (HashMap): ");
 callbackHandler.handle(callbacks);
}

```

4. Obtain the managed connection factory and properties in the login method. For example:

```

// The method getManagedConnectionFactory must be used as shown for compatibility with WAS Classic
ManagedConnectionFactory managedConnectionFactory = ((WSManagedConnectionFactoryCallback) callbacks[0]).getManagedConnectionFactory();
Map properties = ((WSMappingPropertiesCallback) callbacks[1]).getProperties();

```

5. Obtain the user name and password based on the authentication data alias or some other criteria. For example:

```

String alias = (String) properties.get(com.ibm.wsspi.security.auth.callback.Constants.MAPPING_ALIAS);
String user = getUser(alias); // Implementation specific
char[] password = getPassword(alias); // Implementation specific

```

6. Create a `javax.resource.spi.PasswordCredential` object with the user name and password and set the managed connection factory. For example:

```

javax.resource.spi.security.PasswordCredential passwordCredential = new PasswordCredential(user, password);
passwordCredential.setManagedConnectionFactory(managedConnectionFactory);

```

7. Add the password credential to the subject in the commit method. For example:

```

/** {@inheritDoc} */
@Override
public boolean commit() throws LoginException {
 // Verify that the login was successful before adding the PasswordCredential to the subject.
 subject.getPrivateCredentials().add(passwordCredential);
 return true;
}

```

## Developing a programmatic login for obtaining authentication data

8.5.5.9

You can use the Java Authentication and Authorization Service (JAAS) login framework to obtain the authentication data from your application.

### About this task

Your application can perform a JAAS programmatic login using the `DefaultPrincipalMapping` JAAS context entry name to obtain a `Subject` object with a `javax.resource.spi.security.PasswordCredential` instance in the private credentials set that contains the user name and password configured for an `authData` element.

### Procedure

1. Add the `appSecurity-2.0`, `passwordUtilities-1.0`, and `jca-1.7` features in the `server.xml` file. You can also add `appSecurity-2.0`, `passwordUtilities-1.0`, and `jca-1.6`. For example:

```

<featureManager>
 <feature>appSecurity-2.0</feature>
 <feature>passwordUtilities-1.0</feature>
 <feature>jca-1.7</feature>
</featureManager>

```

2. Configure an `authData` element in the `server.xml` file. For example:

```
<authData id="myAuthData" user="myUser" password="myPassword"/> <!-- password can also be encoded -->
```

Encode the password within the configuration. You can get the encoded value by using the `securityUtility` `encode` command.

3. Perform a programmatic login with the `DefaultPrincipalMapping` JAAS login context entry name from your application servlet or enterprise bean, replacing the mapping alias with the one you need. For example:

```
HashMap map = new HashMap();
map.put(com.ibm.wsspi.security.auth.callback.Constants.MAPPING_ALIAS, "myAuthData"); // Replace value with your alias.
CallbackHandler callbackHandler = new com.ibm.wsspi.security.auth.callback.WSMappingCallbackHandler(map, null);
LoginContext loginContext = new LoginContext("DefaultPrincipalMapping", callbackHandler);
loginContext.login();
Subject subject = loginContext.getSubject();
Set<javax.resource.spi.security.PasswordCredential> creds = subject.getPrivateCredentials(javax.resource.spi.security.PasswordCredential.class);
PasswordCredential passwordCredential = creds.iterator().next();
```

**Note:** The error handling is not shown for simplicity. A `javax.security.auth.login.LoginException` is returned if the authentication alias requested does not exist or is malformed.

4. Obtain the user name and password from the `PasswordCredential`. For example:

```
String userName = passwordCredential.getUserName();
char[] password = passwordCredential.getPassword();
// Do something with the userName and password.
```

5. If Java 2 Security is enabled, then the application must be granted the `javax.security.auth.PrivateCredentialPermission`. For example, grant the permission in the application's `META-INF/permissions.xml` file to access the `PasswordCredential` object:

```
<?xml version="1.0" encoding="UTF-8"?>
<permissions xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/xsi.xsd">

 <permission>
 <class-name>javax.security.auth.PrivateCredentialPermission</class-name>
 <name>javax.resource.spi.security.PasswordCredential * "*"</name>
 <actions>read</actions>
 </permission>

 <!-- Other permissions -->

</permissions>
```

For more information about Java 2 Security, see [Liberty: Java 2 Security](#).

## Developing a custom thread identity service

You can develop a custom thread identity service class by implementing the `com.ibm.wsspi.kernel.security.thread.ThreadIdentityService` interface that is provided in the Liberty server. The `ThreadIdentityService` interface is a Service Programming Interface (SPI) that enables support to receive notifications of user identity switches.

### Procedure

1. Create a custom thread identity service by implementing the `ThreadIdentityService` interface.
2. Convert the implementation class into an OSGi service. You can do the conversion in either of two ways:
  - a. Convert your `ThreadIdentityService` class into a Declarative Service (DS) component. For more information, see “Declaring your services to OSGi Declarative Services” on page 1116.
  - b. Write a new `ThreadIdentityService` class that is a DS component and delegate it to your `ThreadIdentityService` class. Register your `ThreadIdentityService` class directly in the Service Registry (SR) by using the OSGi core APIs. For more information, see “Working with the OSGi service registry” on page 1112.



3. Package the custom thread identity service as an OSGi bundle and export the `ThreadIdentityService` service. For information on creating an OSGi bundle, see .
4. Create a feature manifest to include the OSGi bundle. For more information, see “Product extension” on page 577.
5. After the feature is installed into the user product extension location, configure the `server.xml` file with the feature name.

```
<featureManager>
 ...
 <feature>usr:sampleThreadIdentityService-1.0</feature>
</featureManager>
```

---

## Security considerations

Consider the following when you configure Security for Liberty.

### LTPA

- Protect file access to the LTPA keys file because it contains the cryptographic material that is used to encrypt and decrypt the user data. Ensure that only the server and administrators have access to this file.
- Ensure that all servers use the same LTPA keys. In addition, make sure that the all the servers have their time and date synchronized.
- When you specify a password, ensure that it is the same password for all servers that use the same set of LTPA keys. The password is not used to generate the keys, but rather it is used to encrypt the LTPA keys file to prevent the keys from being read. If you copy the LTPA keys file to another Liberty server to achieve Single Sign-On (SSO), the password is required to gain access to the keys in the LTPA keys file. For more information about LTPA, see [Configuring LTPA on Liberty](#) topic.

### Passwords

- Encrypt passwords by using the `securityUtility encode` command.
- If you override the default encryption key with the `wlp.password.encryption.key` property, set the property in a separate configuration file that is stored outside the normal configuration directory for the server.

### Authorization

- If you specify an auth-constraint with no roles in an application, then no one is allowed to access the resource.
- Be cautious when you specify the `EVERYONE` special subject, as this specification is equivalent to not protecting a resource.

### Authentication

- The timeout value for the authentication cache that is specified in the `<authCache>` element must be smaller than the expiration value for the LTPA token that is specified in the `<ltpa>` element.

---

## Securing Liberty by using HTTP Strict Transport Security (HSTS)

You can secure Liberty by first setting up HTTP Strict Transport Security (HSTS) in IBM HTTP Server. Then, add IBM HTTP Server as a front end to Liberty so that connections between Liberty and a client are over HTTPS.

### Procedure

1. Set up HSTS in IBM HTTP Server.
2. Add IBM HTTP Server as a front end to Liberty.



---

## Chapter 8. Developing applications in the Liberty environment

WebSphere Application Server Liberty is a lightweight, composable application server that provides a convenient application development environment for your web and OSGi applications. Applications that run on Liberty also run on the WebSphere Application Server traditional server.

### About this task

Liberty simplifies application development by providing the following key benefits and more:

- Frictionless download, at no cost, for development purposes
- Ultra lightweight modular runtime environment, with an install size of under 50 MB
- Very fast startup time; for example, less than 5 seconds for simple web applications
- Simplified configuration for quick time to productivity
- Java EE and OSGi application deployment support for web applications
- LDAP registry support
- Deployment, as a package, of an application and configured server
- Managed, centralized deployment of a packaged application and server
- Availability of WebSphere Application Server Developer Tools as Eclipse plug-ins for broad tools support
- Platform support for distributed platforms, z/OS, and Mac OS

Very fast restart times, coupled with its small size, dynamic behavior, and ease of use, make Liberty a good option for developers building web applications that do not require the full Java EE environment of traditional enterprise application server profiles. Familiar WebSphere Application Server enterprise qualities of service, such as security and transaction integrity, are enabled as required.

---

## Developing OSGi applications in Liberty

8.5.5.5

8.5.5.7

### About this task

This section includes the OSGi application features that are specific to Liberty. Currently a WebSocket application can be deployed in an OSGi Web Application Bundle on Liberty. For more information on creating an application in an OSGi application on WebSphere Application Server traditional, see [http://www-01.ibm.com/support/knowledgecenter/SSEQTP\\_8.5.5/com.ibm.websphere.wdt.doc/topics/tcrtbundleprj.htm?lang=en](http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wdt.doc/topics/tcrtbundleprj.htm?lang=en).

For information about developing OSGi applications by using WebSphere Developer Tools, see [Developing OSGi applications](#).

## Enable OSGi Applications with Java EE 7 technologies

8.5.5.6

You can enable OSGi Applications with key Java Platform, Enterprise Edition (Java EE) 7 technologies. WebSphere Application Server Liberty Version 8.5.5.6 and later, is a production ready server certified for Java EE 7 Full Platform.

To achieve zero migrations, new features are created and existing features remain unaltered. For example, when support for servlet 3.1 was added, a `servlet-3.1` feature was created, and `servlet-3.0` was kept to ensure behavior did not change for an existing server deployment.

In an environment before Java EE 7 support, you were required to configure `blueprint-1.0`, or something that depends on it, to be able to deploy OSGi applications. You then either configured other OSGi-specific features, like `wab-1.0`, or generic ones, like `jpa-2.0`, to get other capabilities.

In a Java EE 7 environment, OSGi configuration is in two steps:

1. Decide you want to deploy OSGi Bundles.
2. Decide what technologies you want to use to implement those OSGi Bundles.

## Telling the server you want to deploy OSGi Bundles

The first step is to add the `osgiBundle-1.0` feature to your `server.xml`:

```
<featureManager>
 <feature>osgiBundle-1.0</feature>
</featureManager>
```

Adding the `osgiBundle-1.0` feature enables OSGi Bundles to be deployed as part of an OSGi application.

## Telling the server what component models you want to use

Rather than having OSGi-specific features, like `wab-1.0`, you now configure the same component models you would for Java EE. This configuration enables the use of servlets in Bundles, for example, Web Application Bundles or Http Whiteboard servlets:

```
<featureManager>
 <feature>osgiBundle-1.0</feature>
 <feature>servlet-3.1</feature>
</featureManager>
```

For more information, see , , and .

This server configuration adds the ability to use `jpa-2.1` in a Persistence Bundle:

```
<featureManager>
 <feature>osgiBundle-1.0</feature>
 <feature>servlet-3.1</feature>
 <feature>jpa-2.1</feature>
</featureManager>
```

For more information, see .

Optionally, you can still include `blueprint-1.0`:

```
<featureManager>
 <feature>osgiBundle-1.0</feature>
 <feature>servlet-3.1</feature>
 <feature>jpa-2.1</feature>
 <feature>blueprint-1.0</feature>
</featureManager>
```

## Which Java EE 7 component models are supported?

You can see the full list of Java EE 7 technologies that are enabled for OSGi applications.

## Enabling integration of OSGi application services



8.5.5.5

Using the `osgiAppIntegration-1.0` feature, the OSGi applications that are available within the same Java virtual machine (JVM) can share their services with each other. To enable communication between the OSGi applications, you must declare the appropriate service headers in the application manifest file, `META-INF/APPLICATION.MF`. An application that wants to import services from other applications must include the `Application-ImportService` header, and an application that wants to export services to other applications must include the `Application-ExportService` header. If an application wants to export and import services, then both headers must be used.

## About this task

To enable the `osgiAppIntegration-1.0` feature, you must include the feature in the `server.xml` file. To use the feature, you need to use the appropriate headers, such as the `Application-ImportService` and `Application-ExportService`. For more information, see `Application manifest files`. You must add the `binding:=local` directive in the headers to specify the integration of applications that are within the same JVM. The binding directive is specific only to the `osgiAppIntegration-1.0` Liberty feature.

## Procedure

1. Based on your requirement, add one or both application headers to your `MANIFEST.MF` file as given in the following example:

```
Application-ExportService: com.acme.Foo;binding:=local
Application-ImportService: com.acme.Foo;binding:=local
```

where `com.acme.Foo` is the name of the Java interface or class associated with the OSGi service.

**Note:** You must add the `binding:=local` directive along with the application import and export service headers to allow the applications within the same JVM to communicate with each other.

2. Add the feature in the `server.xml` file.  

```
<feature>osgiAppIntegration-1.0</feature>
```

## Custom blueprint namespace handlers

8.5.5.4

The Blueprint Container specification, introduced in the OSGi Enterprise Specification Release 5, provides a simple and easy programming model for creating dynamic applications in the OSGi environment without adding complexity to the Java code.

For more information about the OSGi Enterprise Release specification, see `OSGi specification download`.

The Blueprint Container specification defines a Dependency injection framework for OSGi. It is designed to handle the dynamic nature of OSGi, where services can become available and unavailable at any time. The specification is also designed to work with plain old Java objects (POJOs) so that the same objects can be used within and outside the OSGi framework. The Blueprint XML files that define and describe the various components of an application are key to the Blueprint programming model. The specification describes how the components get instantiated and wired together to form a running application. For more information, see `OSGi Blueprint Container Specification`.

Each blueprint bundle must contain a blueprint XML file in order for the blueprint runtime to process the blueprint component of the bundle. The standard blueprint element is defined by the OSGi blueprint specification, and required in every blueprint xml document. It sets the default document namespace to `http://www.osgi.org/xmlns/blueprint/v1.0.0`, for example:

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
```

Other namespaces can be added to the blueprint by using standard XML rules, either as prefixed entries, or directly within the custom XML elements. These namespaces can be added either at the top level or they can be inline with the custom XML elements. If it is valid XML, it is parsed correctly. For example, defined in the top-level blueprint element:

```
<blueprint
 xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
 xmlns:tx="http://aries.apache.org/xmlns/transactions/v1.0.0">
```

For example, inline in a custom element:

```
<transaction method="*" value="Required"
 xmlns:tx="http://aries.apache.org/xmlns/transactions/v1.0.0"/>
```

The Blueprint runtime implementation that is provided by Apache Aries project is used to support Blueprint bundles that are contained in OSGi Applications for Liberty. For more information, see Apache Aries. The Aries Blueprint runtime provides an extension mechanism called namespace handlers. A namespace handler provides a processor for custom blueprint extensions or namespaces. A namespace handler implements the `org.apache.aries.blueprint.NamespaceHandler` interface and must be registered in the OSGi service registry with an associated `osgi.service.blueprint.namespace` service property. This property denotes the namespace URIs this handler can process. For example: `http://aries.apache.org/xmlns/transactions/v1.0.0`. The service property value can either be a single String or URI, or a Collection, or an array of String or URI.

The blueprint runtime parses the blueprint descriptors twice. The first pass is fast, and finds only every namespace that is used by the blueprint bundle. If the blueprint bundle uses a non-standard namespace, then the blueprint container attempts to locate `NamespaceHandler` services in the OSGi service registry for each custom namespace. A `NamespaceHandler` service advertises every xml namespace that it can process by using OSGi service properties. The blueprint runtime does not parse the blueprint xml until `NamespaceHandler` services can be found for every custom namespace that is used in the bundle. Unless `NamespaceHandler` services can be found for every custom namespace, the blueprint container is unable to process the bundle. This result can mean that the blueprint container waits indefinitely if no `NamespaceHandler` exists. If this situation is encountered, then the blueprint container issues a warning to the log. When the blueprint parser begins to parse the blueprint xml files, it parses any standard blueprint elements. When the parser reaches a custom element, the parser calls out to the `NamespaceHandler` that advertised support for the namespace of the custom element. Here, the `NamespaceHandler` has the opportunity to process the information in the custom element, modify the runtime blueprint model, or do any other operation. If there is a typing error in any of the namespace definitions, then the blueprint almost certainly fails to start.

A custom `NamespaceHandler` service can be provided by any bundle that is running in Liberty, including Liberty Feature bundles , and OSGi Applications bundles.

---

## Developing WebSocket applications in Liberty

You can configure Liberty to use the WebSocket protocol to enable applications to communicate by using a full duplex connection.

### About this task

To configure a Liberty server to run an application that is enabled for WebSocket 1.0, you must set the `websocket-1.0` feature for WebSocket 1.0 or the `websocket-1.1` feature for WebSocket 1.1.

**Note:** 8.5.5.7

In addition to the WebSocket API which is defined in JSR 356 (Java API for WebSocket version 1.1), an API was added to the WebSphere implementation which allows a servlet or filter to request that the current HTTP Request be "upgraded" to start a WebSocket session. This new API is documented here:

## Interface WsWsocServerContainer

WebSocket endpoints can make use of templates to match an endpoint to a URI. URIs that should not map to websocket endpoints, even though they match a websocket template, can also be used by a web application. The distinction between mapping a websocket endpoint to a URI or allowing the URI to be treated as a "non-websocket" HTTP Request, is made by the presence or absence of an "Upgrade" header with a value of "websocket" in the HTTP Request.

For information about developing WebSocket applications by using WebSphere Developer Tools, see Developing WebSocket applications.

## WebSocket



8.5.5.4

WebSocket is a standard protocol that enables a web browser or client application and a web server application to communicate by using one full duplex connection.

HTTP was not designed for long-lived, real-time, full duplex communication between two applications. In many instances, a user's web server application or servlet wants to communicate with a client browser or application in a long-lived, real-time, full duplex conversation. In other words, the two applications want to freely read and write data back and forth. An example of this type of application is one that constantly displays changing currency exchange rates on the web browser of a stock trader. Current solutions that involve existing HTTP technology to accomplish this type of communication are cumbersome and inefficient. HTTP solutions, for constant two-way communication between a browser and a server, mostly consists of either polling or two open HTTP connections that handle one-way traffic only, or both.

WebSocket uses a standard HTTP request-response sequence to establish a connection. When the connection is established, the WebSocket API provides a read and write interface for reading and writing data over the established connection in an asynchronous full duplex manner. WebSocket also provides an interface for asynchronously closing the connection from either side.

Because WebSocket uses a standard HTTP request-response sequence to establish a connection, the connection initiation connects through firewalls and proxies in the same manner as an HTTP connection. WebSocket requires full duplex communication, including simultaneous reads and writes on the same connection. Version 8.5.5.3 and later of the WebSphere web server plug-in support full duplex communication, but other firewalls and proxies might require modification to enable this support. WebSocket can also use SSL for secure connections and transmission of data. This protocol uses SSL in the same way that the HTTP protocol uses SSL.

The Liberty WebSocket feature implements the following specifications:

- The WebSocket Protocol - RFC 6455
- Java API for WebSocket - JSR 356



8.5.5.5

Liberty supports the WebSocket 1.0 and WebSocket 1.1 specifications. Compared to WebSocket 1.0, WebSocket 1.1 supports a more robust way of specifying message handlers.

You can download sample programs that implement the WebSocket protocol from WASdev.net. For a walkthrough of using WebSocket on Liberty, see WebSocket sample application on WASdev.net.





---

## Chapter 9. Deploying applications in Liberty

You can deploy web applications, enterprise applications, and OSGi applications in Liberty. You deploy an application by either dropping the application into a previously-defined dropins directory, or by adding an application entry to the server configuration.

### Before you begin

**Distributed operating systems** You can deploy applications as described in this topic, or as described in “Adding and running an application on Liberty by using developer tools” on page 1331.

This topic assumes that you have not disabled dynamic updates to the runtime configuration, as described in “Controlling dynamic updates” on page 975.

### About this task

By default, the “dropins” directory is automatically monitored. If you drop an application into this directory, the application is automatically deployed on the server. Similarly, if the application is deleted from the directory, the application is automatically removed from the server. The “dropins” directory can be used for applications that do not require additional configuration, such as security role mapping. If you put your applications in the “dropins” directory, you must not include an entry for the application in the server configuration. Otherwise, the server will try to load the application twice and an error might occur. For applications that are not in the “dropins” directory, you specify the location using an application entry in the server configuration. The location can be on the file system or at a URL.

Your application can be packaged as an archive file, a directory, [8.5.5.4](#) or as a loose application where files are in multiple locations. For more information about loose applications, see “Loose applications” on page 1412.

For applications in the “dropins” directory, the file name and file extension are used by the application monitor to determine the type of application, and to generate the application id and application name. For example, if the archive file or directory is named snoop.war, the application monitor assumes that the application is a web application and that the application id and application name is “snoop”. For configured applications, the application type and name are specified.

For more information about the default directory structure and the properties that are associated with directories (for example `server.config.dir`), see “Directory locations and properties” on page 894.

**Note:** Restrictions apply when using the “dropins” directory in a production environment. See “Versioning is not possible for applications in the “dropins” directory” on page 1484.

### Procedure

- Deploy an application by dropping it into the dropins directory.

For example, using the default directory structure, to deploy an application you drop it into the `${server.config.dir}/dropins` directory (that is, `wlp/usr/servers/server_name/dropins`).

You can deploy your application in any of the following ways:

- Place the archive file with its identifying suffix (`.ear`, `.war`, and so on) directly into the `/dropins` directory. For example, `${server.config.dir}/dropins/myApp.war`
- Extract the archive file into a directory named with the application name and the identifying suffix. For example, `${server.config.dir}/dropins/myApp.war/WEB-INF/...`

- Place the archive file or the extracted archive into a subdirectory named with the identifying suffix. For example, `${server.config.dir}/dropins/war/myApp/WEB-INF/...`
- Deploy an application by adding it to the server configuration file.

Configure the application element in the `server.xml` configuration file. See **\*\*\*\* MISSING FILE \*\*\*\***. You must configure the following attributes for the application:

**id** Must be unique and is used internally by the server.

**name** Must be unique and depending on the application. The value of **name** might be used as the context-root of the application. For more information on how the context-root is set for an application, see “Deploying a web application to Liberty” on page 1344.

**type** Specifies the type of application.

- For web applications, the supported type is **war**.
- For enterprise applications, the supported type is **ear**.

**location**

Specifies the location of the application. It can be an absolute path or a URL which you can download the application from. It can also be the file name of your application (including file extension if any).

If the application is available on the file system, the location can either be the full path name or a simple file name. If the location does not include the full path, the application manager looks for the application in `${server.config.dir}/apps` and `${shared.app.dir}`. If the application is available at a URL, the application manager downloads the application to a temporary folder inside the server work area, then starts the application.

**Note:** The location that you specify for a configured application should not be in the “dropins” directory. If you drop an application into the “dropins” directory, and also specify the location in the `server.xml` file, you are telling the server to deploy the application twice.

In the following two examples, the location is the file system. If the location is a URL, enter the URL in the location field.

```
<osgiApplication location="D:/apps/ImpactEBA.eba"/>
<webApplication location="ImpactWeb.war"/>
```

The second example does not include the full path. In this case, you must put the application in one of the following locations:

- `${server.config.dir}/apps` (that is, *server\_directory/user/servers/server\_name/apps*)
- `${shared.app.dir}` (that is, *liberty\_install\_location/usr/shared/apps*)

You can deploy your application to the file system in either of the following ways:

- Place the archive file with its identifying suffix (`.ear`, `.war`, and so on) directly into the chosen location. For example, *application\_directory\_path/myApp.war*
- Extract the archive file into a subdirectory of the chosen location, named with the application name and the identifying suffix. For example, *application\_directory\_path/myApp.war/WEB-INF/...*

**Note:**

- You must create the server-level apps directory, whereas the shared apps directory is present by default. See “Directory locations and properties” on page 894 for more information about the properties associated with the server directories.
- The **application** element can be set before or after the server has started. If the element is set after the server has started, the changes are picked up dynamically.

-  **8.5.5.6** Deploying Contexts and Dependency (CDI) applications in Liberty

A Liberty server can be used to deploy CDI applications by configuring the server for the CDI 1.2 Liberty feature. See “Configuring Liberty for Contexts and Dependency Injection 1.2” on page 1071 for more information.

Applications that use contexts and dependency injection must have CDI enabled. For the CDI 1.2 Liberty feature, CDI is enabled if either:

- There is a `beans.xml` file with a bean discovery mode of `all`.
- There is no `beans.xml` file or a blank `beans.xml` file and classes with bean defining annotations. In this case, there must be a bean deployment archive.

For more information on the different types of bean deployment archive recognized by the CDI 1.2 feature, see “Contexts and Dependency Injection 1.2 behavior changes” on page 1072.

- Remove an application.

For applications that are included in the server configuration, remove the reference to the application from the `server.xml` file. The application is then automatically removed from the server.

For applications that are deployed to the “dropins” directory, delete the application from the directory. The application is then automatically removed from the server.

To uninstall all applications that are in the “dropins” directory, set the application monitor `dropinsEnabled` property to `false` as described in “Controlling dynamic updates” on page 975.

## What to do next

For all deployed applications, you can configure whether application monitoring is enabled and how often to check for updates to applications. For the “dropins” directory, you can also configure the name and location of the directory and choose whether to deploy the applications that are in the directory. See “Controlling dynamic updates” on page 975.

---

## Adding and running an application on Liberty by using developer tools

### Distributed operating systems

You can add applications to the server by right-clicking on the server in the Servers view then selecting Add and Remove from the menu.

### Before you begin

This task assumes that your application is in your Eclipse workspace. If you have a prebuilt application archive file that you want to add and run, you must first import the file into your Eclipse workspace. Alternatively, you can use the steps described in Chapter 9, “Deploying applications in Liberty,” on page 1329.

### About this task

When you add an application to the server, the workbench tries to determine which features are required by the application and enables them in the server configuration for you if they are not already enabled.

### Procedure

1. In the Servers view, right-click the server and select **Add and Remove**.
2. In the **Add and Remove** wizard, under the **Available** list, select the applications you want to add then click **Add**. Or click **Add All** to add all available applications to the server.
3. Alternatively, you can right-click on an application in the **Project Explorer** view and select **Run As > Run on Server**, or **Debug As > Debug on Server**. This adds the application to the server (if not already added), starts the server (if not already started) and runs the application.

## Results

**Tip:** If you are using **Run on Server** or **Debug on Server** and the server is already started, the browser might try to load the application before the server has finished loading it. If this happens, wait for the message in the console view that displays the application has been started and then refresh the browser if necessary.

## Publishing your application by using developer tools

### Distributed operating systems

Publishing involves copying files (projects, resource files, and server configurations) to the correct location for the server to find and use them. You can either publish your application automatically or manually.

### About this task

If you choose to publish your application automatically, the publishing frequency is controlled by a configurable publishing interval. If you do not want to wait for the automatic publishing interval to pass, or the **Never publish automatically** option is enabled, at anytime you can manually request the workbench to issue a publish command to the server. Each manual publish command causes a single publishing request to the server.

### Procedure

- To publish your application to a server automatically, complete the following steps:
  1. Open the Server preferences page by clicking **Window > Preferences > Server > Launching**, and select the **Automatically publish when starting servers** check box. The workbench checks to see if your project and files on the server are synchronized. If they are not, the project and the files are automatically updated when the server is either started or restarted.
  2. (optional) Modify the Publishing settings. To modify the Publishing settings, right-click the server in the Servers view and select **Open**. In the Overview page of the server editor, under the Publishing settings, you will find the following settings:
    - **Never publish automatically:** Specifies the workbench should never publish files to the server.
    - **Automatically publish when resources change:** Specifies the workbench to issue a publish after changes on a file that is associated with the server are saved and a full time interval has passed in the **Publishing interval** setting. In the workbench, the default setting is the **Automatically publish when resources change** option is enabled with a value set in the publishing interval.
    - **Automatically publish after a build event:** Specifies the workbench to issue a publish after changes on a file that requires a build and is associated with the server are saved, and a full time interval has passed in the **Publishing interval** setting.
    - **Publishing interval (in seconds):** Specifies the number of seconds required before the workbench calls a publish to happen on the server. However, if you make a subsequent change to the files before this time interval has completed, the publish is delayed as the timer is reset. The workbench makes a publish to the server only after the full time interval has passed. If you set the publishing interval to 0 seconds, an immediate publish should happen after changes on a file are saved.
- To publish your application to a server manually, complete the following steps:
  1. Select the server and then click the **Publish to the server** icon located on the toolbar.
  2. Right-click the server and then select **Publish**.

## Publishing settings for a WebSphere Application Server Liberty

*Publishing* involves copying files such as application, resource files, and deployment descriptor files to the correct location for the server to find and use them. You can choose whether you want to publish your application on the server or run your application within the development environment without copying the application into the directories of the server.

### About this task

#### Run applications directly from the workspace

The **Run applications directly from the workspace** publishing option requests the server to run your application from the workspace.


This publishing option publishes faster when an application contains a single root, as opposed to containing multiple roots. The workbench might require additional processing time to publish an application with multiple roots. To determine whether the structure of your application contains a single or multiple roots, use the Project Structure Validator. For details, see *Creating and configuring Java EE projects using wizards*.

#### CAUTION:

**When you are using the Run applications directly from the workspace publishing option, the server can lose track of your application under the following scenarios:**

- If you delete your workspace, the server can no longer find your application. As a result, if you did not put your application under source control management and the workspace is deleted, you can lose your application from your file system.
- If you delete an application from the workspace without removing it from the server, the server can no longer find your application. As a result, you might encounter errors when starting the server because the server tries to start the missing application from the workspace.

### Procedure

1. In the Servers view, double-click your WebSphere Application Server Liberty server to open the server editor.
2. In Liberty Settings, configure publishing settings by choosing any or none of the following options:
  - You can select **Run applications directly from the workspace**.  
This option is selected by default.
  -  You can select **Allow publishing of applications containing errors**.  
If you do not select this option, then when you attempt to publish an application that contains errors, you receive a warning. You can choose to cancel publishing and fix your errors, or you can choose to publish your application without correcting the errors. If you select this option, then you do not receive a warning when you attempt to publish an application that contains errors.
3. Save and close the editor.

## Restart requirements for a modified application on Liberty

When you change an application on Liberty, whether an application or server restart is required depends on whether the Liberty server is started in run mode or debug mode.

### Modifying an application when the server is started in debug mode

In debug mode, some changes require an application restart. For details of which changes require an application restart, use the IBM Rational Application Developer documentation, and see the “Hot Method Replace” section in WebSphere Application Server debug limitations.

If an application restart is required, a Hot Code Replace Failed dialog box is displayed, giving you an option to restart the application. If you click **Yes**, the application is restarted automatically; if you click **No**, you must restart the application manually.

For details on how to start the server in debug mode, use the IBM Rational Application Developer documentation, and see [Debug on server](#).

## Modifying an application when the server is started in run mode

In run mode, most changes require an application restart. If an application restart is required, the application is restarted automatically as part of the application publish operation from the developer tools. Some changes require a server restart.

---

## Customizing automatic feature detection

Control which features are added to each project and resolve feature conflicts.

### About this task

Automatic feature detection selects and enables features that an application needs. However, the automatic process can lead to conflicts among features. Customize automatic feature detection to fit your preferences and resolve conflicts.

### Procedure

1. Right-click the project and select **Properties**.
2. Expand **Liberty** and select **Required Features**.
3. Set the action for each feature to **Always add**, **Prompt before adding**, or **Never add**. **Always add** is the default. See [Disabling automatic feature detection](#) to set the default for all projects or to disable automatic feature detection.
4. Resolve or ignore feature conflicts by using the Feature Conflict dialog box.
  - a. The Feature Conflict dialog box appears when features conflict. New features are marked with **new**.
  - b. Click **Add...** to add a feature, or select a feature and click **Remove** to remove it.
  - c. Click **OK** after you resolve the conflicts.

**Note:** Click **Ignore** to avoid conflict resolution. If conflicts are ignored, the server might not work as expected.

---

## Packaging a Liberty server from the command line

From the command line, you can create a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server. You can also choose to exclude the runtime binary files from the compressed file.

### About this task

The Liberty server is lightweight, and therefore you can easily package a server installation in a compressed file. You can store this package, distribute it to colleagues, and then use it to deploy the installation to a different location or to another machine, or even embed the installation in a product distribution.

The server installation that you want to package cannot already be joined to a collective. You can only package a stand-alone server.

**Note:** **Distributed operating systems** The resulting file is created by using UTF-8 encoding for entry names, so the tool that you use to open the file must be able to use UTF-8 encoding for entry names. The `jar` command in a Java SDK uses this format.

## Procedure

To package a Liberty server from the command line, complete the following steps:

1. Open a command line, then change directory to the `wlp/bin` directory.
2. Stop the server.
3. Run the **package** command to create a package.

**8.5.5.5** You can package the Liberty profile server or the runtime.

- Package the Liberty server.

The default archive format is `.zip` on all platforms apart from z/OS where it is `.pax`. You can also generate a `.jar` archive.

If you do not specify a server name, `defaultServer` is used. If you do not specify the **--archive** parameter, the value of `server_name` is used for `package_file_name`, and the compressed file is created in the `${server.output.dir}` directory.

Choose the correct command for your environment.

**Distributed operating systems**

IBM i

Use this command to generate a `.zip` archive.

```
server package server_name --archive=package_file_name.zip --include=all
```

where `package_file_name.zip` is a file name that you choose. This file name can include a full path name. If the full path is omitted, a compressed file called `package_file_name.zip` is created in the `${server.output.dir}` directory.

**Distributed operating systems**

IBM i

Use this command to generate a `.jar` archive. The advantage of a `.jar` archive is that the scripts in the `bin` directory keep their permissions, so they are executable when the package is installed.

```
server package server_name --archive=package_file_name.jar --include=all
```

where `package_file_name.jar` is a file name that you choose.

For more information about extraction options with this archive file, see “Java archive file extraction options” on page 837.

You can also use the **--include** option with this command. For example, the `--include=all` option packages the runtime binaries and the relevant files in the `${WLP_USER_DIR}` directory; the `--include=usr` option packages only relevant files in the `${WLP_USER_DIR}` directory, effectively excluding the runtime binaries from the compressed file.

The `--include=usr` option is not valid with an archive format of `.jar`.

If you use the `--include=minify` option, the **server** command packages only those parts of the runtime environment, and files in the `${WLP_USER_DIR}` directory, that are required to run the server. This option significantly reduces the size of the resulting archive.

The parts of the runtime environment that are retained by the `minify` operation depend on the features that are configured in the server that you are packaging. Only those features that are required to run the server are retained, and the remaining features are removed. Therefore, you cannot later enable a feature that has been removed. For example, if only the `servlet-3.0` feature is retained, you cannot later enable the `jpa-2.0` feature.

You can repeat the `minify` operation to further reduce the size of the archive if the configuration is changed. There is, however, no reverse operation for the `minify` operation, so if you later require one or more features that have been removed, you must begin again with a complete Liberty server.

While the minify operation is running, the server is temporarily started, and you see the associated messages. For this reason, you cannot use the `--include=minify` option with a server that is not able to be started, but you can package it with the `--include=all` or `--include=usr` options.

You can specify the operating systems that you want the packaged server to support by using the `--os` option with the `--include=minify` option.

For example, to package a server with z/OS support removed, use the following command:

```
server package --archive="nozoz.zip" --include=minify --os=-z/OS
```

To package a server with OS/400 support retained, but z/OS support removed, use the following command:

```
server package --archive="small.zip" --include=minify --os=OS/400,-z/OS
```

To package a server that supports only Linux, use the following command:

```
server package --archive="linux.zip" --include=minify --os=Linux
```

- **8.5.5.5** Package the Liberty runtime.

Create a runtime archive that contains the `wlp` directory, but does not contain the `usr` directory. The naming convention for a server package is *package\_name.zip*; for example, *CustomerPortalApp.zip*. To create a runtime archive, run the **package** command without a server name and with the `--include=wlp` option:

```
server package --include=wlp
```

To specify a package file name and target location, add the `--archive=package_path_name` option; for example:

```
server package --include=wlp --archive=c:\temp\myPackage.zip
```

If you do not specify a valid package name or target location with the `--archive` option, then the command creates the `wlp.zip` runtime archive in the `$WLP_OUTPUT_DIR` location, which is the `${wlp.install.dir}/usr/servers` directory by default. The target location must exist before running the command. Thus, if the target location is `c:\temp`, the `C:\temp` directory must exist and have write permission for the command to write the archive to the `C:\temp` directory.

---

## Using JNDI binding for constants from the server configuration files

You can bind constants into the default Java Naming and Directory Interface (JNDI) namespace from the server configuration files by using the `<jndiEntry>` element on Liberty.

### About this task

The default JNDI namespace is available in the Liberty profile to provide bindings to miscellaneous objects required by applications. Any data sources declared in the server configuration files are available in the default JNDI namespace. Additionally, you can bind Java strings and primitive data types in the configuration file into JNDI namespace. These constants are then made available to an application at run time, providing a simple and portable way to pass configuration values into the application.

For more information about the JNDI naming, see Naming.

### Procedure

1. Add a constant into the default JNDI namespace by specifying the `jndi-1.0` Liberty feature in the `server.xml` file of the Liberty server.

```
<featureManager>
 <feature>jndi-1.0</feature>
</featureManager>
```

2. Bind constants into the JNDI namespace by specifying the `<jndiEntry>` elements with **jndiName** and **value** attributes in the `server.xml` file.

```
<jndiEntry jndiName="schoolOfAthens/defaultAdminUserName" value="plato" />
<jndiEntry jndiName="schoolOfAthens/defaultAdminPassword" value="republic" />
```



If you want to bind an instance of `java.net.URL` into the JNDI namespace, use the `jndiURLEntry` configuration:

```
<jndiURLEntry jndiName="urls/IBMKnowledgeCenter" value="http://www-01.ibm.com/support/knowledgecenter/" />
<jndiURLEntry jndiName="urls/WASDevNet" value="http://wasdev.net" />
```

3. Look up the constants from an application by using a JNDI context with the following code:

```
Object jndiConstant = new InitialContext().lookup("schoolOfAthens/defaultAdminUserName");
String defaultAdmin = (String) jndiConstant;
```

**Note:**

- The `lookup()` method returns an object to the application. The type of the object is determined by interpreting the value stored in the `jndiEntry` element as a Java literal string or primitive data type. If the parsing fails, the exact value is provided as an unmodified string.
- The `jndiEntry` element supports the integer, floating-point, boolean, character, and string literals as described in Java Language Specification, Java SE 7 Edition, section 3.10. String and character literals might contain unicode escaped sequences (see section 3.3: Unicode Escaped Sequences ), and the octal and character escape sequences ( see section 3.10.6: Escape Sequences for Character and String Literals). Null literals and class literals are not supported; for more information see section 3.10.7: The null literal and section 15.8.2: Class Literals .

See the following examples of Java literals:

- The string "Hello, world" followed by a newline character:  

```
<jndiEntry jndiName="a" value="'Hello, world.\n'" />
```
- The integer with a binary value 1010101:  

```
<jndiEntry jndiName="b" value="0b1010101" />
```
- The single character 'X':  

```
<jndiEntry jndiName="c" value="'X'" />
```
- The double-precision floating point number 1.0:  

```
<jndiEntry jndiName="d" value="1.0D" />
```

For more information about `<jndiEntry>` element, see \*\*\*\* MISSING FILE \*\*\*\*.

---

## Using JNDI binding for dynamic values from the server configuration files

8.5.5.4

You can bind a reference for dynamic values into the default Java Naming and Directory Interface (JNDI) namespace from the server configuration files by using the `jndiReferenceEntry` element on the Liberty profile.

### About this task

The default JNDI namespace is available in Liberty to provide bindings to miscellaneous objects that are required by applications. Based on the features that are enabled in your server, you can bind a predetermined set of objects to the default JNDI namespace. Additionally, you can bind a reference to an object factory, which dynamically determines the value that it returns. You can also use this object factory to return custom object types to an application.

For more information about the JNDI naming, see Naming.

### Procedure

1. Add the `jndi-1.0` Liberty feature to the `server.xml` file.

```
<featureManager>
 <feature>jndi-1.0</feature>
</featureManager>
```

2. Create an ObjectFactory class that returns a programmatically defined value.

```
import javax.naming.spi.ObjectFactory;

public class MyObjectFactory implements ObjectFactory {
 @Override
 public Object getObjectInstance(Object o, Name n, Context c, Hashtable<?, ?> envmt) throws Exception {
 Properties p = new Properties();
 p.put("abc", 123);
 return p;
 }
}
```

3. Include the ObjectFactory in a library element in the server.xml file:

```
<library id="objectFactoryLib">
 <fileset dir="${server.config.dir}/lib" includes="factory.jar"/>
</library>
```

4. Declare the factory in a jndiObjectFactory element in the server.xml file and reference the previously declared library.

```
<jndiObjectFactory id="objectFactory" libraryRef="objectFactoryLib"
 className="com.ibm.example.factory.MyObjectFactory"/>
```

You can also declare the type of object that the factory returns. The type is returned by the `javax.naming.Context.list()` method.

```
<jndiObjectFactory id="objectFactory" libraryRef="objectFactoryLib"
 className="com.ibm.example.factory.MyObjectFactory"
 objectClassName="java.util.Properties"/>
```

5. Declare the entry in a jndiReferenceEntry element in the server.xml file and reference the previously declared factory.

```
<jndiReferenceEntry id="refEntry" jndiName="ref/entry" factoryRef="objectFactory"/>
```

6. To declare more properties for the jndiReferenceEntry element in the server.xml file:

```
<jndiReferenceEntry id="refEntry" jndiName="ref/entry" factoryRef="objectFactory">
 <properties abc="123"/>
</jndiReferenceEntry>
```

These additional properties are represented as `javax.naming.StringRefAddr` on the `javax.naming.Reference` that is passed to the factory:

```
import javax.naming.spi.ObjectFactory;

public class MyObjectFactory implements ObjectFactory {
 @Override
 public Object getObjectInstance(Object o, Name n, Context c, Hashtable<?, ?> envmt) throws Exception {
 Properties p = new Properties();
 Reference ref = (Reference) o;
 RefAddr refAddr = ref.get("abc");
 p.put("abc", refAddr == null ? 123 : refAddr.getContent());
 return p;
 }
}
```

7. You can inject the resulting object to an application by using a resource environment reference:

```
@Resource(name="ref/entry")
private Properties properties;
```

---

## Deploying OSGi applications to Liberty

You can deploy OSGi applications to Liberty by enabling a list of server features in the server.xml file.

## About this task

By providing a list of OSGi-specific server features, Liberty provides OSGi support for your applications. These features are as follows:

- blueprint-1.0
- osgi.jpa-1.0
- wab-1.0

For a full list of server features in Liberty, see “Liberty features” on page 483.

## Sharing common OSGi bundles for Liberty

You can share common OSGi bundles by placing them in a directory and configuring the `server.xml` file for your server, so that those common OSGi bundles are available to your OSGi applications.

### Procedure

- Create a directory in your file system and place all the common OSGi bundles into the directory.
- Add the following lines into the `server.xml` file.

```
<bundleRepository>
 <fileset dir="directory_path" include="*.jar"/>
</bundleRepository>
```

Where *directory\_path* is the path to the directory that contains the common OSGi bundles.

- Define a dependency on the common bundle using `import` phrase in the `manifest.mf` file of your OSGi application.

---

## Deploying data access applications to Liberty

Deploying a data access application includes more than installing your web application archive (WAR) or enterprise archive (EAR) file onto Liberty. Deployment can include tasks for configuring the data access resources of the server and overall runtime environment.

## About this task

This following topics are covered in this section:

### Procedure

- Configure a data source and JDBC driver for database connectivity in a Liberty profile
- Deploy an JDBC application to Liberty
- Optional: Configure connection pooling in Liberty
- Optional: Develop an application-defined data source on Liberty
- Optional: Configure transaction recovery for data sources on Liberty
- Migrating data access applications to Liberty

## Deploying an existing JDBC application to Liberty

You can take an existing application that uses Java Database Connectivity (JDBC) and a data source, and deploy the application to a server.

## About this task

You can take an existing JDBC application and deploy it to Liberty. To complete this deployment, you add the `jdbc-4.0` Liberty feature to the `server.xml` file. You must also add code that tells the server the JDBC driver location and specifies properties that the JDBC driver uses to connect to the database.

In this example, you can extend your servlet application, or use the one provided here to test the interactivity that is used through the JDBC driver is working as expected.

## Procedure

1. Create a server.
2. Start the server.
3. Add the jdbc-4.0 and the servlet-3.0 Liberty features to the server.xml file.

```
<server>
 <featureManager>
 <feature>jdbc-4.0</feature>
 <feature>servlet-3.0</feature>
 </featureManager>
</server>
```

To check that the server is working and that the features are enabled successfully, see the console.log file, which is stored in the logs directory of the server. You can view it using any text editor. You should see something like this example:

```
[AUDIT] CWWKF0012I: The server installed the following features: [jdbc-4.0, jndi-1.0].
[AUDIT] CWWKF0008I: Feature update completed in 0.326 seconds.
```

4. Specify the database type and the data source location in the server.xml file.

Where *path\_to\_derby* is the location where derby is installed on your operating system, *lib* is the folder where derby.jar is located, and *data/exampleDB* is the directory that is created if it does not exist.

For example:

```
<jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib"/>

<library id="DerbyLib">
 <fileset dir="C:/path_to_derby/lib" includes="derby.jar"/>
</library>

<dataSource id="ds1" jndiName="jdbc/exampleDS" jdbcDriverRef="DerbyEmbedded">
 <properties.derby.embedded
 databaseName="C:/path_to_derby/data/exampleDB"
 createDatabase="create"
 />
</dataSource>
```

For information about other options for coding data source definitions, see “Using Ref tags in configuration files” on page 972.

5. Add some SQL create, read, update, and delete statements to your JDBC application to test the interactivity with the database.

```
package wasdev;

import java.io.*;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.annotation.Resource;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import javax.sql.DataSource;

@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {

 @Resource(name = "jdbc/exampleDS")
 private DataSource ds1;
```

```

private Connection con = null;
private static final long serialVersionUID = 1L;

public HelloWorld() {
 super();
}
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<H1>Hello World Liberty</H1>\n");
 try {
 con = ds1.getConnection();
 Statement stmt = null;
 stmt = con.createStatement();
 // create a table
 stmt.executeUpdate("create table cities (name varchar(50) not null primary key, population int, county var
 // insert a test record
 stmt.executeUpdate("insert into cities values ('myHomeCity', 106769, 'myHomeCounty')");
 // select a record
 ResultSet result = stmt.executeQuery("select county from cities where name='myHomeCity'"); would result.ne
 // display the county information for the city.
 out.println("The county for myHomeCity is " + result.getString(1));
 // drop the table to clean up and to be able to rerun the test.
 stmt.executeUpdate("drop table cities");
 }
 catch (SQLException e) {
 e.printStackTrace();
 }
 finally {
 if (con != null){
 try{
 con.close();
 }
 catch (SQLException e) {
 e.printStackTrace();
 }
 }
 }
}
}
}

```

## 6. Compile your application.

Where *path\_to\_liberty* is the location you installed Liberty on your operating system, and *path\_to\_app* is the location of the Java file of the application you want to compile.

Example on Windows:

```

C:\> javac -cp
path_to_liberty\wlp\dev\api\spec\com.ibm.ws.javaee.servlet.3.0_1.0.1.jar
path_to_App\HelloWorld.java

```

Example on Linux:

```

mo@machine01:~> javac -cp
path_to_liberty/wlp/dev/api/spec/com.ibm.ws.javaee.servlet.3.0_1.0.1.jar
path_to_App/HelloWorld.java

```

If the **javac** command is not recognized, ensure that you have the Java bin directory in the PATH environment variable of your operating system.

## 7. Add the application to the server.

In this example, the JDBC application is put in the dropins directory of the server:

```

... \dropins\HelloWorldApp.war\WEB-INF\classes\wasdev\HelloWorld

```

The wasdev directory uses the same package name that is used in HelloWorld.java.

## 8. Check that your JDBC application is working.

For this example, go to this URL:

`http://localhost:9080/HelloWorldApp/HelloWorld`

Port 9080 is the default HTTP port that is used by the Liberty server. You can check which HTTP port your server is set on by looking in the `server.xml` file.

The output on the browser for this example looks like:

```
Hello World Liberty
The county for myHomeCity is myHomeCounty
```

## Enabling JDBC Tracing for Liberty

JDBC tracing for Liberty is enabled either through a driver-specific custom trace setting, or using the application server supplemental JDBC tracing option.

### About this task

There are two ways of using driver-specific custom trace facilities:

- Using the Java built-in logging mechanism, `java.util.logging`, if the driver supports it.
- Configuring a custom trace setting as a vendor property.

If your JDBC driver does not provide its own custom tracing or logging facilities, or the facilities it provides are minimal, you can use supplemental JDBC tracing from the application server.

If you enable tracing by using either a custom vendor property or supplemental JDBC tracing, you must add the logwriter name to the trace specification in the `bootstrap.properties` file. You can use any of the following logwriters:

**DB2** `com.ibm.ws.db2.logwriter`

**Derby** `com.ibm.ws.derby.logwriter`

**Informix JCC (uses the same driver as DB2)**  
`com.ibm.ws.db2.logwriter`

**Informix JDBC**  
`com.ibm.ws.informix.logwriter`

**Microsoft SQL Server JDBC Driver**  
`com.ibm.ws.sqlserver.logwriter`

**DataDirect Connect for JDBC for Microsoft SQL Server**  
`com.ibm.ws.sqlserver.logwriter`

**Sybase**  
`com.ibm.ws.sybase.logwriter`

**Other databases (for example solidDB and MySQL)**  
`com.ibm.ws.database.logwriter`

Because changes to trace enablement involve altering the `bootstrap.properties` file, you must restart the server for the changes to take effect.

The following examples illustrate the use of the various JDBC trace methods.

### Procedure

- Use `java.util.logging`.

If the driver you are using supports `java.util.logging`, you can enable it by appending the driver's trace level to `com.ibm.ws.logging.trace.specification` in the `bootstrap.properties` file. See [Using Java logging in an application](#), and the JDBC vendor documentation for levels and other trace information specific to your driver.

Here is an example for Microsoft SQL Server JDBC Driver:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.microsoft.sqlserver.jdbc=FINE
```

Here is an example for Oracle JDBC:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:oracle=FINE
```

- For Oracle, you must also enable the tracing using the system property `oracle.jdbc.Trace`, using one of the following two options:

- In the `bootstrap.properties` file, add the setting `oracle.jdbc.Trace=true`

- In a Java program, add the setting `System.setProperty("oracle.jdbc.Trace","true");`

- Use custom trace settings.

If the driver you are using has custom trace settings, you set them as JDBC driver vendor properties in the `server.xml` file. You also add the logwriter name to the trace specification in the `bootstrap.properties` file.

Here is an example for DB2 JCC, using the custom property `traceLevel`:

- Example code for the `server.xml` file:

```
<dataSource id="db2" jndiName="jdbc/db2" jdbcDriverRef="DB2Driver" >
 <properties.db2.jcc databaseName="myDB" traceLevel="-1"/>
</dataSource>
```

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.db2.logwriter=all=enabled
```

Here is an example for Derby Network Client:

- Example code for the `server.xml` file:

```
<dataSource id="derbyNC" jndiName="jdbc/derbyNC" jdbcDriverRef="DerbyNC" >
 <properties.derby.client databaseName="myDB" createDatabase="create" traceLevel="1"/>
</dataSource>
```

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.derby.logwriter=all=enabled
```

Here is an example for Informix JCC. This database uses the DB2 drivers for JCC connectivity.

- Example code for the `server.xml` file:

```
<dataSource id="informixJCC" jndiName="jdbc/informixJCC" jdbcDriverRef="InformixDriverJCC" >
 <properties.informix.jcc databaseName="myDB" traceLevel="-1"/>
</dataSource>
```

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.db2.logwriter=all=enabled
```

- Use supplemental JDBC tracing.

If your JDBC driver does not provide suitable tracing or logging facilities, you can use supplemental JDBC tracing from the application server. The application server automatically determines whether to enable supplemental JDBC tracing, based on the JDBC driver being used. To override this, set the data source property `supplementalJDBCTrace` to true or false.

#### 1. Enable supplemental tracing.

Here is an example for enabling supplemental tracing with the embedded Derby database.

Supplemental JDBC tracing is enabled by default for this database, so you only need to set the logwriter in the `bootstrap.properties` file:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.derby.logwriter=all=enabled
```

Here is an example for enabling supplemental tracing with Informix JDBC. Supplemental JDBC tracing is enabled by default for this database.

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.informix.logwriter=all=enabled
```

Here is an example for enabling supplemental tracing, and `java.util.logging`, with Microsoft SQL Server JDBC Driver:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.sqlserver.logwriter=all=enabled:
com.microsoft.sqlserver.jdbc=all
```

Here is an example for enabling supplemental tracing with DataDirect Connect for JDBC for Microsoft SQL Server:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.microsoft.sqlserver.jdbc=all
```

Here is an example for enabling supplemental tracing with solidDB. Supplemental JDBC tracing is enabled by default for this database.

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.database.logwriter=all=enabled
```

Here is an example for enabling supplemental tracing with Sybase. Supplemental JDBC tracing is enabled by default for this database.

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.sybase.logwriter=all=enabled
```

Here is an example for enabling supplemental tracing with other databases:

- Example code for the `bootstrap.properties` file:

```
com.ibm.ws.logging.trace.specification==audit=enabled:com.ibm.ws.database.logwriter=all=enabled
```

## 2. Disable supplemental tracing

To disable supplemental JDBC tracing, either set the `supplementalJDBCTrace` data source property to `false` in the `server.xml` file, or remove the `logwriter` name from the `com.ibm.ws.logging.trace.specification` property in the `bootstrap.properties` file:

- Example code for the `server.xml` file for solidDB:

```
<dataSource id="soliddb" jndiName="jdbc/soliddb"
 jdbcDriverRef="solidDBDriver" supplementalJDBCTrace="false">
 <properties databaseName="dba" URL="jdbc:solid://localhost:2315/dba/dba" />
</dataSource>
```

- Example code for the `bootstrap.properties` file for solidDB:

```
com.ibm.ws.logging.trace.specification==audit=enabled
```

**Note:** If you are not seeing JDBC trace, a feature might be activating JDBC immediately. Check `bootstrapping.properties` and edit it to add JDBC trace specifications.

---

## Deploying a web application to Liberty

By deploying a `helloworld.war` application, you can learn how server configurations change in Liberty.

### Before you begin

The `helloworld.war` application uses a simple servlet to display a message on your browser. You can create any other messages to be displayed. The coding of the application is not described within Liberty documents.

### About this task

When you deploy a web application to Liberty by using the developer tools, all configurations that are related to the application are automatically enabled in the `server.xml` file. However, you can also configure the `server.xml` file manually by completing the following steps.



This example uses the `helloworld.war` application and can be accessed by using `http://localhost:9090/helloworld`. In this example, a Liberty server instance is created, and then its default HTTP port is changed to 9090, and then an application is deployed to it.

## Procedure

1. Create a server named `hwserver` by using the command `server create hwserver`.
2. Copy the `helloworld.war` application into the `/usr/servers/hwserver/apps` directory; this directory was created by the `server create` command in step 1.
3. In the `server.xml` file that was created by the `server create` command, change the default HTTP port of the server `hwserver` to 9090 by replacing the attribute value `httpPort="9080"` with `httpPort="9090"`:

```
<server description="new server">

 <!-- Enable features -->
 <featureManager>
 <feature>jsp-2.2</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint"
 host="localhost"
 httpPort="9090"
 httpsPort="9443" />
</server>
```

4. Configure the application by updating the `server.xml` in either of the following ways:

- Define the application by using a `webApplication` element:

```
<server description="Hello World Server">

 <featureManager>
 <feature>servlet-3.0</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9090" />

 <webApplication contextRoot="helloworld" location="helloworld.war" />
</server>
```

- Define the application by using an `application` element:

```
<server description="Hello World Server">

 <featureManager>
 <feature>servlet-3.0</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9090" />

 <application context-root="helloworld" type="war" id="helloworld"
 location="helloworld.war" name="helloworld"/>
</server>
```

The `webApplication` element can use the same child elements as the `application` element, except for **context-root** and `type`. The two elements do not work together for a **context-root**, and if both an `application` and `webApplication` element define the same **context-root**, only one is used and an error is displayed.

The **context-root** attribute specifies the entry point of the deployed application. The entry point of a deployed application is determined in the following precedence:

- **context-root** in the `server.xml` file
- `application.xml`, if an EAR application
- `ibm-web-ext.xml`, if a web application

- **name** of the application in the `server.xml` file, if a web application
- `Manifest.MF`, if a WAB application
- Directory name or the file name relative to the drop-ins directory of Liberty

**Note:** In an application `server.xml` configuration, the `application` element can contain a **context-root** tag. This **context-root** tag is applicable in combination with the tag `type="war"`. For all other application types, the **context-root** element has no effect.

It is not possible to override the **context-root** for either an EAR application, or an EBA application. It is only possible to do an override for a stand-alone war file, or webApplication.

5. Start the server in foreground by using the command `server run hwserver`.
6. Test the application at `http://localhost:9090/helloworld`.
7. Optional: Stop the server if you don't need it.

---

## Deploying SIP applications to Liberty

8.5.5.7

Session Initiation Protocol (SIP) applications are Java applications that contain at least one SIP servlet. SIP applications are deployed the same way as other web applications.

### Before you begin

Install the `sipServlet-1.1` feature in your Liberty server. For more information, see “Adding and removing Liberty features” on page 968.

Configure the SIP container.

### About this task

To deploy a SIP application, the application must be packaged in a web archive (WAR) file, a servlet archive (SAR) file, or an enterprise archive (EAR) file that contains a WAR or SAR file.

This task describes how to manually deploy a SIP application. Alternatively, you can deploy a SIP application to Liberty by using the developer tools, which automatically enable all configurations related to the application in the `server.xml` file. For more information, see [WebSphere Developer Tools > Developing > Developing SIP applications](#).

### Procedure

Add your SIP application WAR, SAR, or EAR file to the Liberty server in one of the following ways:

- Move the archive file to the folder for drop-in artifacts in the server configuration directory at `wlp/usr/servers/server_name/dropins`. The Liberty server monitors the `dropins` folder for new applications and automatically installs the application with the default configuration.
- Move the archive file to the folder for applications in the server configuration directory at `wlp/usr/servers/server_name/apps`. Then, install the SIP application in the Liberty server by configuring an `application` element in the `server.xml` file.

The following example installs the `appName.ear` file. The `context-root` attribute specifies the entry point of the deployed application.

```
<application id="appId" name="appName" type="ear" location="appName.ear" context-root="/sip289"/>
```

---

## Deploying a JPA application to Liberty

To enable Liberty to support an application that uses the Java Persistence API (JPA), you add the `jpa-2.0` or `jpa-2.1` feature to the `server.xml` file, depending on which specification level you need. You also need to define persistence contexts and persistence units, and configure access to the entity manager and entity manager factory.

### Before you begin

This task assumes that you have created a Liberty server, on which you want to deploy an application that uses JPA. See “Creating a Liberty server manually” on page 883.

### About this task

**8.5.5.6** There are two JPA features available in Liberty:

- The `jpa-2.0` feature provides support for applications that use application-managed and container-managed JPA written to the JPA 2.0 specification. The support is built on Apache OpenJPA with extensions to support the container-managed programming model.
- The `jpa-2.1` feature provides support for applications that use application-managed and container-managed JPA written to the JPA 2.1 specification. The support is built on EclipseLink

For information about developing JPA applications by using WebSphere Developer Tools, see *Developing JPA applications*.

### Procedure

- Add the `jpa-2.0` or `jpa-2.1` feature to the `server.xml` file.
- Add persistence context and persistence unit definitions to the `web.xml` file.

For example:

```
<persistence-context-ref>
 <persistence-context-ref-name>example/em</persistence-context-ref-name>
 <persistence-unit-name>ExamplePersistenceUnit</persistence-unit-name>
</persistence-context-ref>
```

```
<persistence-unit-ref>
 <persistence-unit-ref-name>example/emf</persistence-unit-ref-name>
 <persistence-unit-name>ExamplePersistenceUnit</persistence-unit-name>
</persistence-unit-ref>
```

- Configure access to the entity manager.

For example:

```
Context ctx = new InitialContext();
UserTransaction tran = (UserTransaction) ctx.lookup("java:comp/UserTransaction");
tran.begin();
EntityManager em = (EntityManager) ctx.lookup("java:comp/env/example/em");
Thing thing = new Thing();
em.persist(thing);
tran.commit();
```

- Configure access to the entity manager factory.

For example:

```
Context ctx = new InitialContext();
EntityManagerFactory emf = (EntityManagerFactory) ctx.lookup("java:comp/env/example/emf");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Thing thing = new Thing();
```

```
em.persist(thing);
tx.commit();
int id = thing.getId();
em.close();
```

## Enhancement of JPA entities

The JPA 2.0 specification provider that is included in Liberty is based on Apache OpenJPA. OpenJPA uses Java bytecode enhancement of JPA persistent types (Entity, Embeddable, MappedSuperclass) to add state tracking, and other necessary information to enable persistence and other optimized features within JPA classes. In an application server environment, enhancement of your JPA entities occurs automatically when the application is loaded by the Liberty server.

Pre-enhancement of JPA classes (or build time enhancement) is necessary when a persistence JAR is used in both application server, and non-application server environments. The most common ways to perform build time enhancement are the OpenJPA enhancer Ant task, and PCEnhancer. These build time enhancement options require the OpenJPA library and dependent libraries to be on the classpath. The **wsenhancer** command, in the WebSphere Application Server traditional install, can also be used.

**Note:** The JPA 2.1 specification provider for Liberty is EclipseLink. EclipseLink does not require entity enhancement.


---

## Deploying web services applications to Liberty

By configuring Liberty features in the `server.xml` file, you can deploy web services applications to Liberty.

## Deploying JAX-RS 2.0 applications to Liberty

You can use Java API for RESTful Web Services (JAX-RS) to develop services that follow Representational State Transfer (REST) principles. RESTful services are based on manipulating resources. Resources can contain static or dynamically updated data. By identifying the resources in your application, you can

make the service more useful and easier to develop. Liberty provides two Liberty features, `jaxrs-1.1`  and `jaxrs-2.0`, to support the JAX-RS programming model.

8.5.5.6

## Asynchronous processing

You can use the asynchronous processing technique in JAX-RS 2.0 to process threads. Asynchronous processing is supported both in the Client API and in the Server API. For more information about asynchronous processing in Client and Server APIs, see Chapter 8 of JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services (the "Specification").

The following two examples show asynchronous processing in the Client and Server APIs:

- Asynchronous processing in the Client API:

```
Client client = ClientBuilder.newClient();
WebTarget target = client.target("http://example.org/customers/{id}");
target.resolveTemplate("id", 123).request().async().get(
 new InvocationCallbackCustomer() {
 @Override
 public void completed(Customer customer) {
 // Do something
 }
 @Override
 public void failed(Throwable throwable) {
 // Process error
 }
 });
```

- Asynchronous processing in the Server API:

```

@Path("/async")
public class MyResource{

 @GET
 public void getAsync(@Suspended final AsyncResponse asyncResponse){
 CompletionCallback callBack = new CompletionCallback(){
 @Override
 public void onComplete(Throwable throwable) {
 ...
 }
 };
 asyncResponse.register(callBack);
 asyncResponse.resume("some Response");
 }
}

```

The JAX-RS 2.0 implementation in Liberty supports EJB and the use of stateless and singleton session beans as root resource classes. When an EJB method is annotated with `@Asynchronous`, the EJB container automatically allocates the necessary resources for its execution. Thus, in this scenario, it is unnecessary to use an `Executor` to generate an asynchronous response. For example,

```

@Stateless
@Path("/")
class EJBResource {

 @GET @Asynchronous
 public void longRunningOp(@Suspended AsyncResponse ar) {
 executeLongRunningOp();
 ar.resume("Hello async world!");
 }
}

```

Explicit thread management is not needed in this case because that is under the control of the EJB container. The response is produced by calling `resume` on the injected `AsyncResponse`. Hence, the return type of `longRunningOp` is `void`.

## | **Configuring a resource to receive multipart/form-data parts from an HTML form submission in JAX-RS 2.0**

| HTML forms that transmit file data must be configured with the POST method and the "multipart/form-data" action. This data can be received in one of the two ways by the JAX-RS resource method that accepts it with the IBM Java API for RESTful Web Services (JAX-RS) implementation.

### | **About this task**

| This task provides instructions for configuring a JAX-RS method to use and produce multipart/form-data. The following example illustrates an HTML form:

```

| <form action="http://www.example.com/" method="POST" enctype="multipart/form-data">
| <input type="text" name="fileid" />
|

| <input type="text" name="description" />
|

| <input type="file" name="thefile" />
|

| <input type="submit" name="submit" value="submit"/>
| </form>

```

| You can implement the IBM JAX-RS to receive the data in parts, so you can process these parts yourself, if needed.

### | **Procedure**

| Create a resource method. You must declare one of the following resource methods to receive and echo multipart/form-data content from an HTTP POST:

```

package com.example.jaxrs;
@POST
@Consumes("multipart/form-data")
@Produces("multipart/form-data")

public Response postFormData(IMultipartBody multipartBody) {
 List<IAttachment> attachments = multipartBody.getAllAttachments();
 String formElementValue = null;
 InputStream stream = null;
 for (Iterator<IAttachment> it = attachments.iterator(); it.hasNext()); {
 IAttachment attachment = it.next();
 if (attachment == null) {
 continue;
 }
 DataHandler dataHandler = attachment.getDataHandler();
 stream = dataHandler.getInputStream();
 MultivaluedMap<String, String> map = attachment.getHeaders();
 String fileName = null;
 String formElementName = null;
 String[] contentDisposition = map.getFirst("Content-Disposition").split(";");
 for (String tempName : contentDisposition) {
 String[] names = tempName.split("=");
 formElementName = names[1].trim().replaceAll("\\\\", "");
 if ((tempName.trim().startsWith("filename"))) {
 fileName = formElementName;
 }
 }
 if (fileName == null) {
 StringBuffer sb = new StringBuffer();
 BufferedReader br = new BufferedReader(new InputStreamReader(stream));
 String line = null;
 try {
 while ((line = br.readLine()) != null) {
 sb.append(line);
 }
 } catch (IOException e) {
 e.printStackTrace();
 } finally {
 if (br != null) {
 try {
 br.close();
 } catch (IOException e) {
 e.printStackTrace();
 }
 }
 }
 formElementValue = sb.toString();
 System.out.println(formElementName + ":" + formElementValue);
 } else {
 //handle the file as you want
 File tempFile = new File(fileName);
 ...
 }
 }
 if (stream != null) {
 stream.close();
 }
 return Response.ok("test").build();
}

```

Or

```

package com.example.jaxrs;
@POST
@Consumes("multipart/form-data")
@Produces("multipart/form-data")

public Response postFormData(List<IAttachment> attachments) {
 List<IAttachment> attachments = multipartBody.getAllAttachments();
 String formElementValue = null;
 InputStream stream = null;
 for (Iterator<IAttachment> it = attachments.iterator(); it.hasNext()); {
 IAttachment attachment = it.next();
 if (attachment == null) {
 continue;
 }
 DataHandler dataHandler = attachment.getDataHandler();
 stream = dataHandler.getInputStream();
 MultivaluedMap<String, String> map = attachment.getHeaders();
 String fileName = null;
 String formElementName = null;
 String[] contentDisposition = map.getFirst("Content-Disposition").split(";");
 for (String tempName : contentDisposition) {
 String[] names = tempName.split("=");
 formElementName = names[1].trim().replaceAll("\\\\", "");
 if ((tempName.trim().startsWith("filename"))) {
 fileName = formElementName;
 }
 }
 }
}

```

```

 if (fileName == null) {
 StringBuffer sb = new StringBuffer();
 BufferedReader br = new BufferedReader(new InputStreamReader(stream));
 String line = null;
 try {
 while ((line = br.readLine()) != null) {
 sb.append(line);
 }
 } catch (IOException e) {
 e.printStackTrace();
 } finally {
 if (br != null) {
 try {
 br.close();
 } catch (IOException e) {
 e.printStackTrace();
 }
 }
 }
 formElementValue = sb.toString();
 System.out.println(formElementName + ":" + formElementValue);
 } else {
 //handle the file as you want
 File tempFile = new File(fileName);
 ...
 }
 }
 if (stream != null) {
 stream.close();
 }
 return Response.ok("test").build();
}

```

The originator of the form POST submission can generate a Content-Transfer-Encoding header for one or more parts of the multipart message. The IBM JAX-RS implementation attempts to auto-decode the payload of the part according to this header when the header is of base64 or quoted-printable encoding type.

## Results

You have received and echoed data from an HTTP POST with multipart/form-data Content-Type, by allowing the IBM JAX-RS implementation to split and auto-decode the parts for you, and by receiving the still encoded parts to process yourself.

## Configuring JAX-RS 2.0 client



8.5.5.6

For Java API for XML RESTful Web Services 2.0, you can configure the client to access REST endpoints. JAX-RS 2.0 introduces a new and standardized Client API so that you can make HTTP requests to your remote RESTful web services.

### About this task

An instance of `Client` is required to access a Web resource using the Client API. The default instance of `Client` can be obtained by calling `newClient` or `build` on `ClientBuilder`.

### Procedure

1. Enable the `jaxrsClient-2.0` or `jaxrs-2.0` feature in your `server.xml` file:

```
<featureManager>
```

```
 <feature>jaxrs-2.0</feature>// If you only need the JAX-RS 2.0 client feature, you can enable jaxrsClient-2.0 ins
```

```
</featureManager>
```

2. Create a JAX-RS 2.0 client and send the request to the server:

```
javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
```

```
javax.ws.rs.client.Client c = cb.build();
```

```
String res = null;
```

```
try {
```

```

res = c.target("<Resource_URL>")
 .path("<PATH>")
 .request()
 .get(String.class);
} catch (Exception e) {
 res = "[Error]:" + e.toString();
} finally {
 c.close();
}

```

For more information about the asynchronous JAX-RS 2.0 client, see “Asynchronous processing” on page 1348.

## What to do next

- 
8.5.5.6 Use the *com.ibm.ws.jaxrs.client.timeout* client property to set the timeout value.


```

javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
 cb.property("com.ibm.ws.jaxrs.client.timeout", "1000");
 Client c = cb.build();

```

**Tip:** The value of the timeout property is milliseconds, and the type must be long or int. If the type of the value is invalid, the following message is displayed:

CWWKW0700E: The timeout value {0} that you specified in the property com.ibm.ws.jaxrs.client.timeout on the JAX-RS Client is invalid.

- 
8.5.5.7 Use the *com.ibm.ws.jaxrs.client.connection.timeout* client property and the *com.ibm.ws.jaxrs.client.receive.timeout* client property to set the timeout values.

- *com.ibm.ws.jaxrs.client.connection.timeout*

```

javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
 cb.property("com.ibm.ws.jaxrs.client.connection.timeout", "1000");
 Client c = cb.build();

```

- *com.ibm.ws.jaxrs.client.receive.timeout*

```

javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
 cb.property("com.ibm.ws.jaxrs.client.receive.timeout", "1000");
 Client c = cb.build();

```

**Tip:** The value of the timeout property is millisecond, and the type must be long or int. If the type of the value is invalid, the following message is displayed:

CWWKW0700E: The timeout value {0} that you specified in the property com.ibm.ws.jaxrs.client.receive.timeout on the JAX-RS Client is invalid.

- Use the following client properties for client proxy support:

```

ClientBuilder cb = ClientBuilder.newBuilder();
cb.property("com.ibm.ws.jaxrs.client.proxy.host", "hostname");
cb.property("com.ibm.ws.jaxrs.client.proxy.port", "8888");
cb.property("com.ibm.ws.jaxrs.client.proxy.type", "HTTP");

```

```

Client c = cb.build();

```

- *com.ibm.ws.jaxrs.client.proxy.host*

- *com.ibm.ws.jaxrs.client.proxy.port*

**Tip:** The type of the proxy server port value must be int. The default value is 80. If the value type is invalid, the following message is displayed:

CWWKW0701E: The proxy server port value {0} that you specified in the property com.ibm.ws.jaxrs.client.proxy.port on the JAX-RS Client is invalid.

- *com.ibm.ws.jaxrs.client.proxy.type*

**Tip:** The value of the proxy server type must be HTTP or SOCKS. The default value is HTTP. If the type of the proxy server is invalid, the following message is displayed:

CWWKW0702E: The proxy server type value {0} that you specified in the property com.ibm.ws.jaxrs.client.proxy.type on the JAX-RS Client is invalid.



- Use the `com.ibm.ws.jaxrs.client.ltpa.handler` client property to set the SSO cookie and set the value to true.

```
ClientBuilder cb = ClientBuilder.newBuilder();
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");
```

If you want to use the Secure Sockets Layer (SSL) function in JAX-RS 2.0, you need to enable the `ssl-1.0` or `appSecurity-2.0` feature. For the LTPA token function, you must enable the `appSecurity-2.0` feature.

For more information about how to configure the environment to run the JAX-RS 2.0 client with SSL through IHS, see [Configuring IBM HTTP server SSL support](#).

**Note:** The `ssl-1.0` feature is a subfeature of the `appSecurity-2.0` feature. If you enable the `jaxrsClient-2.0` feature and the `ssl-1.0` feature, the `appSecurity-2.0` feature is enabled automatically.

- Use the `com.ibm.ws.jaxrs.client.ssl.config` client property to set the SSL reference id of your `server.xml` file.

```
ClientBuilder cb = ClientBuilder.newBuilder();
cb.property("com.ibm.ws.jaxrs.client.ssl.config", "mySSLRefId");
Client c = cb.build();
```

For more information about establishing trust by extracting the certificate from the IHS key file and adding it to the Liberty JKS file, see [Create a key database file and certificates needed to authenticate the Web server during an SSL handshake](#).

**Note:** The configuration in the `server.xml` file shows as follows:

```
<ssl id="mySSLRefId" keyStoreRef="clientKeyStore" trustStoreRef="clientTrustStore" />
```

-  Use the `com.ibm.ws.jaxrs.client.disableCNCheck` client property to disable the common name check.

```
ClientBuilder cb = ClientBuilder.newBuilder();
cb.property("com.ibm.ws.jaxrs.client.disableCNCheck", true);
```

## Deploying EJB in an EAR file for JAX-RS 2.0



8.5.5.6

In Liberty, JAX-RS 2.0 supports EJB JAX-RS in EJB JAR file that must be included in an EAR file.

### Procedure

1. Deploy the new `myearfile.ear` file to Liberty.
2. Use the following URL pattern to access the JAX-RS service::  
`http://<host>:<port>/<context root>/<path of jaxrs resource>`

For example, you can access EJB JAX-RS in `<myejbjaxrs.jar>`:


```
http://<host>:<port>/myejbjaxrs/<path of jaxrs resource>
```

**Note:** If there are EJB JAX-WS classes in the same EJB-jar and the JAX-WS 2.2 feature is enabled, that means JAX-WS router module also exists, then the default context root should be the short file name of EJB jar+ ".jaxrs" like "myejbjaxrs.jaxrs" for `<myejbjaxrs.jar>`.

## Implementation of JAX-RS 2.0 web applications

You can use Java API for RESTful Web Services (JAX-RS) to develop services that follow Representational State Transfer (REST) principles. Using JAX-RS, development of RESTful services is simplified.

JAX-RS is a Java API for developing REST applications quickly. This standard API continues to gain support throughout the Java community. While JAX-RS provides a faster way of developing web

applications than servlets, the primary goal of JAX-RS is to build RESTful services. `jaxrs-1.1` and 

8.5.5.6

**8.5.5.6** jaxrs-2.0 define a server-side component API to build REST applications. IBM JAX-RS provides an implementation of the JAX-RS (JSR 311) specification.

By using the principles of REST, your business applications can benefit from several advantages. RESTful services are generally simpler to develop and consume. Most RESTful services use well-defined standards for delivery such as HTTP. Because HTTP is a protocol that has RESTful properties, RESTful services gain scalability advantages that enable the service to serve different clients and interoperate with multiple services, while permitting future growth. Additionally, clients for RESTful services generally are not difficult to develop, yielding interoperability advantages because most RESTful services use common data representations such as XML and JSON.

By using JAX-RS technology, REST applications are simpler to develop, simpler to consume, and simpler to scale when compared to other types of distributed systems. Many popular and widely used Internet services have successfully provided RESTful APIs to their applications. Third parties have used various REST APIs to build their own businesses and applications.


JAX-RS capabilities are provided by the use of a servlet or a filter. When you configure the web.xml file of your web application and assemble the IBM JAX-RS implementation that is based on the Apache Wink framework into the library directory of your web application, your business application is now ready to use JAX-RS capabilities.

For more information, see

- Define the resources in JAX-RS web applications
- Configure the JAX-RS application
- Assemble JAX-RS web applications
- Deploy JAX-RS web applications

**Note:** The context root value in Liberty is either the name of the web module, or the user-defined context root found in the EAR file.

## Implementation of secure JAX-RS applications

The JAX-RS 1.1 runtime environment from IBM is driven by a servlet derived from the Apache Wink project.  **8.5.5.6** The JAX-RS 2.0 runtime environment is driven by a servlet derived from the Apache CXF 3.0.2. Within the WebSphere Application Server environment, the lifecycle of servlets is managed in the web container. Therefore, the security services offered by the web container are applicable to REST resources that are deployed in WebSphere Application Server.

You can define and add security constraints on the REST resources using the same tools that is used to assemble REST applications. These constraints are captured in the J2EE web deployment descriptor that is associated with your application. The following list describes security definitions that you can include in the deployment descriptor:

- User authentication when invoking REST resources embodied in the application, including
  - HTTP basic authentication.
  - Form login authentication.
- Authorization control over REST resources as defined by the URL patterns for the resources.
- Use of SSL for transport when invoking REST resources.
- Programmatic use of the SecurityContext object to determine user identity and roles.

All the security mechanisms supported by the web container are applicable to REST resources, including the use of the Kerberos-based SPNEGO authentication mechanism.

For more information, see:

- Securing JAX-RS applications within the web container

- Securing JAX-RS resources using annotations
- Securing downstream JAX-RS resources

**Note:** In Liberty, the default context root is the name of the WAR file. For more information about options when configuring context roots, see “Deploying a web application to Liberty” on page 1344.

### Securing downstream JAX-RS resources:

You can secure downstream Java API for RESTful Web Services (JAX-RS) resources by configuring the BasicAuth method for authentication and by using the LTPA JAX-RS security handler to take advantage of single sign-on for user authentication.

### Before you begin

This task assumes that you have completed the following steps:

- You have installed your JAX-RS application onto the application server.
- You have enabled security for your JAX-RS application.
- You have secured your JAX-RS applications within the web container by configuring downstream JAX-RS applications to use the basic authentication (BasicAuth) method for user authentication.

### About this task

When composing JAX-RS resources, a new LTPA JAX-RS security handler can be used to seamlessly authenticate on downstream resource invocations.

When invoking downstream secure JAX-RS resources, the calling application is required to authenticate to the target resource. If the target resource on a downstream server uses the BasicAuth method for security, the calling application can take advantage of single sign-on (SSO) for JAX-RS resources. Using single sign-on, an authenticated context is propagated along downstream calls. You can use the LTPA-based security client handler to authenticate to downstream resources that are distributed across servers.

To illustrate this scenario, assume that you have two servers in your cell and that you have deployed JAX-RS resources on both of these servers. Suppose from one resource on server1 you need to invoke another resource that is deployed on server2. When server2 resources are secured using the BasicAuth method for authentication, use the LTPA JAX-RS security handler to take advantage of single sign-on and seamlessly propagate user authentication on downstream calls without having to provide or manage user identities and passwords in the application.

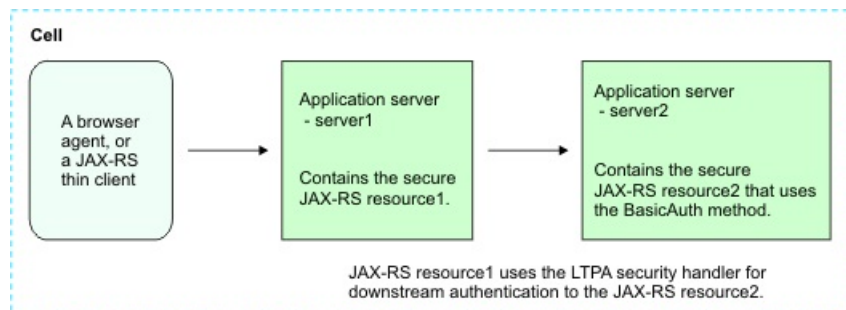



Figure 19. Securing JAX-RS downstream resources

Use the following steps to configure user authentication to a downstream server using the JAX-RS security handler at application build time.


## Procedure

1. At application build time, use the LTPA-based security client handler, `LtpaAuthSecurityHandler`, to authenticate to downstream resources that are distributed across servers.
  - For JAX-RS 1.1, when using the `LtpaAuthSecurityHandler` class, ensure that you target resources using the `https` scheme for your URLs, and that the target application is SSL-enabled. It is highly recommended to use SSL connections when sending user credentials, including LTPA cookies. You may explicitly turn off the requirement for SSL in the `LtpaAuthSecurityHandler` class by invoking the `setSSLRequired` method on the security handler with the `false` value. The default value is `true`.  
`yourLtpaAuthSecHandler.setSSLRequired(false);`

-  **8.5.5.6** For JAX-RS 2.0, you can use the `com.ibm.ws.jaxrs.client.ltpa.handler` client property to set SSO cookie and set the value to `true`:

```
ClientBuilder cb = ClientBuilder.newBuilder();
```

```
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");
WebTarget t = c.target("http://" + serverIP + ":" + serverPort + "/" + moduleName + "/ComplexClientTest/ComplexResource");
String res = t.path("echo1").path("test1").request().get(String.class);
c.close();
ret.append(res);
```

-  **8.5.5.6** If you want to use the Secure Sockets Layer (SSL) function in JAX-RS 2.0, you need to enable the `ssl-1.0` or `appSecurity-2.0` feature. For the LTPA token function, the `appSecurity-2.0` feature is must.

**Note:** The `ssl-1.0` feature is a subfeature of the `appSecurity-2.0` feature. If you enable the `jaxrsClient-2.0` feature and the `ssl-1.0` feature, the `appSecurity-2.0` feature is enabled automatically.

2. Add the security handler to the handlers chain.
3. Create the REST client instance.
4. Create the resource instance that you want to interact with.
5. Substitute a value representing your resource address.

## Results

You have defined secure JAX-RS resources such that when downstream resources are invoked, you can use single sign-on and seamlessly propagate user authentication on downstream calls without having to provide or manage user identities and passwords in the application.

## Example

For JAX-RS 1.1, the following code snippet demonstrates how to use this security handler that is packaged as part of the JAX-RS client.

```
import org.apache.wink.client.Resource;
import org.apache.wink.client.RestClient;
import org.apache.wink.client.ClientConfig;
import org.apache.wink.client.handlers.LtpaAuthSecurityHandler;

ClientConfig config = new ClientConfig();
LtpaAuthSecurityHandler secHandler = new LtpaAuthSecurityHandler();

// Add this security handler to the handlers chain.
config.handlers(secHandler);

// Create the REST client instance.
RestClient client = new RestClient(config);

// Create the resource instance that you want to interact with.
// Substitute a value representing your resource address
```

```
resource =
 client.resource("http://localhost:8080/path/to/resource");

// Now you are ready to begin calling your resource.
```



8.5.5.6

For JAX-RS 2.0, the following code snippet demonstrates how to use this security handler that is packaged as part of the JAX-RS client.

```
ClientBuilder cb = ClientBuilder.newBuilder();
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");

String res = "";
res = c.target("http://" + serverIP + ":" + serverPort + "/" + moduleName + "/rest/ltpa")
 .request()

c.close();
return res;
```

## JAX-RS 2.0 behavior changes



8.5.5.6

The JAX-RS 2.0 implementation contains some behavior changes. These changes might cause applications to behave differently or fail on JAX-RS 2.0 if the applications are upgraded from JAX-RS 1.1.

The following list describes the differences between JAX-RS 1.1 and JAX-RS 2.0:

- In JAX-RS 1.1 and Jersey, if an EJB or CDI class creates a new instance that is returned by the JAX-RS application.getSingletons() method, the engine uses the returned instance and does not try to access the instance from the EJB or CDI container. In JAX-RS 2.0, for the same scenario, the engine tries to access the instance from the EJB or CDI container. If the instance can be accessed, the retrieved instance is used. But if the instance cannot be accessed, the returned instance from the getSingletons() method is used. For example:

```
@Override
public SetObject getSingletons() {
 SetObject objs = new HashSetObject();
 objs.add(new CDIInjectResource());
 objs.add(new EJBInjectResource());
 return objs;
}
```

- JAX-RS 2.0 includes many API changes when it handles the MultiPart file. For example, in JAX-RS 1.1, the @FormParam can be used to handle the MultiPart file, but in JAX-RS 2.0, only @MultiPartBody or @Attachment can be used to handle the MultiPart file. For more information, see Configuring a resource to receive multipart/form-data parts from an HTML form submission in JAX-RS 2.0.
- The jackson packages that are displayed as a third-party API in JAX-RS 1.1 are no longer displayed in JAX-RS 2.0. If you want to use any org.codehaus.jackson APIs in your application, you need to compress the jackson packages in your application.
- If you specify javax.ws.rs.core.Application for the servlet name in the web.xml file, the getClasses method in the Application object, which is injected by @Context, does not return the resource classes.

```
<servlet>
 <servlet-name>javax.ws.rs.core.Application</servlet-name>
</servlet>
<servlet-mapping>
 <servlet-name>javax.ws.rs.core.Application</servlet-name>
 <url-pattern>/*</url-pattern>
</servlet-mapping>
```

- The JAX-RS 2.0 specification states that a provider is a class that implements one or more JAX-RS interfaces and that can be annotated with @Provider for automatic discovery. In the scenario, a class has @Local annotation that refers to a provider interface, but it does not implement any POJO provider interface, and then it is an invalid provider. For example:

```
@Stateless
@Local(OneLocalInterfaceMyOtherStuffMessageBodyWriter.class)
public class OneLocalInterfaceMyOtherStuffProvide
```

- If you use the `MessageBodyReader` and `MessageBodyWriter` `@Consumes` and `@Produces` annotations, some supported media types might be restricted. Use the `isReadable` method or `isWriteable` method to check the media type. For example:

```
@Provider
@Consumes("<custom/type>")
@Produces("<custom/type>")
@Singleton
public class MyMessageBodyReaderAndWriter implements MessageBodyReader,MessageBodyWriter {

 public boolean isReadable(Class<?> type,
 Type genericType,
 Annotation[] annotations,
 MediaType mediaType) {
 if (mediaType.toString().equals("<custom/type>"))
 return true;
 return false;
 }

 public boolean isWriteable(Class<?> type,
 Type genericType,
 Annotation[] annotations,
 MediaType mediaType) {
 if (mediaType.toString().equals("<custom/type>"))
 return true;
 return false;
 }

 ...
}
```

- You can use asynchronous processing in JAX-RS 2.0 to process threads. For more information, see “Asynchronous processing” on page 1348.
- None of the Wink APIs that are displayed as third-party APIs in JAX-RS 1.1 are supported in JAX-RS 2.0. Here is a partial list:
  - `org.apache.wink.common.model.atom.AtomEntry`. For more information about integrating JAX-RS 2.0 with Atom, see “JAX-RS 2.0 integration with Atom” on page 1360.
  - `org.apache.wink.client.handlers.BasicAuthSecurityHandler`. If you want to use basic authentication in JAX-RS 2.0, see the following code snippets:

1. Use `ClientRequestFilter` through the JAX-RS 2.0 standard Client API as shown in the code example:

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import javax.ws.rs.client.ClientRequestContext;
import javax.ws.rs.client.ClientRequestFilter;
import javax.ws.rs.core.MultivaluedMap;
import javax.xml.bind.DataConverter;

public class BasicAuthFilter implements ClientRequestFilter {

 private final String usr;
 private final String pwd;

 public BasicAuthFilter(String usr, String pwd) {
 this.usr = user;
 this.pwd = pwd;
 }

 public void filter(ClientRequestContext requestContext) throws IOException {
 MultivaluedMap<String, Object> headers = requestContext.getHeaders();

 String token = this.usr + ":" + this.pwd;
```

```

 final String basicAuthentication = "Basic " + DatatypeConverter.printBase64Binary(token.getBytes("UTF-8"));
 headers.add("Authorization", basicAuthentication);
 }
}

```

## 2. Register to the ClientBuilder:

```

ClientBuilder cb = ClientBuilder.newBuilder();
cb.register(new BasicAuthFilter("user", "password"));

```

- org.apache.wink.client.handlers.LtpaAuthSecurityHandler. If you want to use the LTPA-based security client to secure downstream resources, see “Securing downstream JAX-RS resources” on page 1355.
- org.apache.wink.server.internal.providers.exception.EJBAccessExceptionMapper. This API is no longer supported because it is the Wink-specified ExceptionMapper. You can define your own ExceptionMapper to map the EJBAccessException.
- com.ibm.websphere.jaxrs.server.IBMRestFilter. This API is no longer supported because it is based on Wink Filter.

**Note:** Detect if there are wink jar packages in your application. If there are any wink packages in your application, you must do the following steps:

1. Make sure that there is Application subclass defined.
2. At least one of getClasses and getSingletons must not return null.

- For more information about the supported client properties that can be used in the JAX-RS 2.0 client, see Configuring JAX-RS 2.0 client.
- If you want to use the Secure Sockets Layer (SSL) function in JAX-RS 2.0, do the following steps:
  1. Enable either the ssl-1.0 feature or the appSecurity-2.0 feature. For the LTPA token function, the appSecurity-2.0 feature is required.

**Note:** The ssl-1.0 feature is a sub-feature of the appSecurity-2.0 feature. If you enable the jaxrsClient-2.0 feature and the ssl-1.0 feature, the appSecurity-2.0 feature is automatically enabled.

2. Enable the com.ibm.ws.jaxrs.client.ssl.config property in the JAX-RS 2.0 client code as follows:

```

ClientBuilder cb = ClientBuilder.newBuilder();
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ssl.config", "mySSLConfig"); //mySSLConfig is the ssl ref id in Liberty server

```

**Note:** This property can bind the Liberty SSL configuration to scopes of ClientBuilder, Client, and WebTarget.

- If you want to use the Wink Client in the JAX-RS 2.0 server run time, do the following steps:
  1. Download the following files that can enable Wink Client in the JAX-RS 2.0 server run time.
    - Download the Apache Wink and related JAR files from <http://wink.apache.org/downloads.html>.
    - Download the Apache HTTP and related JAR files from <http://hc.apache.org/>.

**Note:** If the JAX-RS 2.0 feature is not enabled, you must also download and add the JAX-RS API to the third-party lib. Download the JAX-RS API from <https://jax-rs-spec.java.net/nonav/>.

2. Save all JAR files into the <third-party lib> directory.
3. Add the location of <third-party lib> to the server.xml file:

```

<library id="thirdPartyLib">
 <fileset dir=" <third-party lib>" includes="*.jar" scanInterval="5s"/>
</library>
<enterpriseApplication id="<Your Ear ID>" location="<Your Ear Name>" name="<Your Ear Name>">
 <classloader commonLibraryRef="thirdPartyLib"/>
</enterpriseApplication>

```

**Note:** For more information about asynchronous processing in Client and Server APIs, see Chapter 8 of JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services (the "Specification").

## JAX-RS 2.0 integration with Atom

8.5.5.8

JAX-RS 2.0 can use Apache Abdera to add Atom support.

You can register the following Apache Abdera-based providers with a JAX-RS endpoint and use resource methods to explicitly deal with Abdera Feed or Entry classes:

- Apache Abdera-based Feed provider: `net.wasdev.wlp.sample.abdera.jaxrs.atom.AtomFeedProvider`
- Apache Abdera-based Entry provider: `net.wasdev.wlp.sample.abdera.jaxrs.atom.AtomEntryProvider`

**Note:** Both `AtomFeedProvider` and `AtomEntryProvider` support a formatted Output property.

Now the JAX-RS 2.0 sample on GitHub supports both Maven and Gradle. The online instruction on GitHub also shows you how to use commands or WebSphere Development Tools (WDT) for Eclipse to build or test Liberty sample. For more information, see <https://github.com/WASdev/sample.abdera.jaxrs>.

**Note:** Apache CXF offers you other ways of integrating JAX-RS 2.0 with Atom. For more information, see <https://cxf.apache.org/docs/jax-rs-data-bindings.html#JAX-RSDataBindings-Atom>.

## JAX-RS 2.0 integration with EJB and CDI



8.5.5.6

JAX-RS 2.0 in Liberty integrates with Enterprise JavaBeans (EJB) and Contexts and Dependency Injection (CDI).

For JAX-RS 2.0 to work with enterprise beans, you need to use `@Path` to annotate the class of a bean and convert it to a root resource class.

By integrating with EJB, you can annotate the EJB beans to expose them as REST endpoints. You can also use the JTA and security functions of EJB. JAX-RS 2.0 in Liberty supports the use of stateless and singleton session beans as root resource classes, providers, and application subclasses. By integrating with CDI, you can annotate CDI beans or Managed beans as REST endpoints and use CDI injection for web services. JAX-RS 2.0 in Liberty supports CDI-style beans as root resource classes, providers, and application subclasses. Providers and application subclasses must be singletons or use the application scope. CDI specification makes it easier to integrate Java EE components of different types. It provides a common mechanism to inject component such as EJB components or Managed beans into other components such as JSPs or other EJBs.

For EJB, you can use annotation with stateless session beans and singleton POJO beans.

- For a stateless session bean, use the `@Stateless` annotation as is shown in the following example:

```
@Stateless
@Path("stateless-bean")
public class StatelessResource {...}
```

- For a singleton bean, use the `@Singleton` annotation as is shown in the following example:

```
@Singleton
@Path("singleton-bean")
public class SingletonResource {...}
```

For CDI, you can use the `@ApplicationScoped` and `@Inject` annotations with application scoped beans.



**Tip:** If the CDI feature is disabled, JAX-RS reports no errors, but the instances are obtained by using POJO.

```
@ApplicationScoped
@Path("/ApplicationScopedResource")
public class ApplicationScopedResource {

 private @Inject
 SimpleBean injected;

 ...
}
```

## Restrictions on JAX-RS 2.0 with EJB and CDI

See the following items for the restrictions of JAX-RS 2.0 in Liberty:

- If you use EJB as JAX-RS resource, provider or application, you cannot use the `@Context` injection on constructor of the EJB bean. The reason is that the EJB with default constructor can only be used for JAX-RS according to EJB and JAX-RS specification.
- If you use EJB or CDI annotation in a Java class, but the Liberty feature for EJB (such as `ejbLite-3.2`) or CDI (such as `cdi-1.0`) is not configured in the `server.xml` file, which means there are no EJB or CDI supports in Liberty run time, then the JAX-RS 2.0 engine uses the Java class as POJO class.
- For an Application class, if it implements no interface or it has `@LocalBean` annotation, it is seen as EJB; if it implements local or POJO interfaces, it is not seen as EJB.
  - For a Provider:
    - If a class implements POJO provider interfaces only without the `@Local` annotation, it is seen as a valid EJB provider.
    - If a class has the `@LocalBean` annotation and it implements the POJO provider interface, then it is seen as a valid EJB provider.
    - If a class has the local interface with the `@Local` annotation, the local interface is a provider interface. If this class implements the provider interface, then it is a valid EJB provider.
    - If a class has a local interface with `@Local` annotation, and if the local interface is not a provider interface, then it is not a valid provider.

The reason is that in this case, EJB container can generate EJB stub for the local interface only rather than the POJO provider interface.
    - If a class has the `@Local` annotation only that refers a provider interface, but it does not implement this provider interface, then it is not a valid provider according to the JAX-RS 2.0 specification: A provider is a class that implements one or more JAX-RS interfaces that are introduced in this specification and that might be annotated with `@Provider` for automatic discovery.
  - For Resource:
    - If EJB-based resource does not implement any interface, all of the methods that are declared in this class are available as JAX-RS resources.
    - If EJB-based resource implements one interface (local or POJO), then all the methods that are declared in this interface are available as JAX-RS resources.
    - If EJB-based resource implements multiple interface,
      1. If all the interfaces are POJO interfaces without the `@Local` annotation, then all the methods that are declared in interface are available as JAX-RS resources.
      2. If all the interfaces are local interfaces with the `@Local` annotation, then all the methods that are declared in the interface are available as JAX-RS resources.

3. If some of the interfaces are local interfaces with the `@Local` annotation while others are not local interfaces, then only the methods declared in the local interfaces are available as JAX-RS resources. The reason is that the EJB container can generate EJB stub for local interfaces only in this scenario.
  4. If the EJB-based resource has the `@LocalBean` annotation, then all the methods that are declared in class are available as JAX-RS resource.
  5. If EJB-based resource implements an interface, then JAX-RS resource method must be declared in the interface. If the interface is a provider that can't be modified, then you must create a new interface for the resource class to add the resource method. Otherwise, it is not seen as EJB resource.
- If a resource class with the `@Path` annotation implements JAX-RS provider interface or it declares with the `@Provider` annotation, this class works as both a resource and a provider. In this case, by default, the JAX-RS 2.0 engine uses only one instance of this class that is shared by the resource and the provider, and the lifecycle of the instance is singleton.
  - If a class is registered in both the `getClasses` and `getSingletons` methods of the application class, then by default, the JAX-RS 2.0 engine uses the instance from the `getSingletons` method and ignore the registration in the `getClasses` method.
  - If a RESTful resource is also a CDI managed bean and its scope is `javax.enterprise.context.Dependent`, the `PreDestroy` method cannot be called because of the CDI restriction.

## JAX-RS 2.0 bean and EJB bean lifecycle

JAX-RS bean and EJB bean have different lifecycle. If the bean lifecycle of JAX-RS and EJB conflicts, the lifecycle is managed by EJB container in Liberty. So the EJB instance is applied while the JAX-RS lifecycle does not work. For more information, see the following table:

Table 94. JAX-RS 2.0 bean and EJB bean lifecycle

Application	JAX-RS 2.0	EJB	Result
Resource	perRequest	Stateless	Stateless
	perRequest	Singleton	Singleton
	Singleton	Stateless	Stateless
	Singleton	Singleton	Singleton
Provider	Singleton	Stateless	Stateless
	Singleton	Singleton	Singleton

## JAX-RS 2.0 scope and CDI scope lifecycle

Beans have a scope that determines the lifecycle of its instances. JAX-RS and CDI have slightly different scopes. If the cope lifecycle of JAX-RS and CDI conflict, see the following table for the result:

Table 95. JAX-RS 2.0 scope and CDI scope lifecycle

Application	JAX-RS 2.0 Scope	CDI Scope annotation	Result
Resource	perRequest	<code>@ApplicationScoped</code>	Singleton
	perRequest	<code>@RequestScoped</code>	perRequest
	perRequest	<code>@Dependent</code>	perRequest
	perRequest	<code>@SessionScoped</code>	Session
	perRequest		perRequest
	Singleton	<code>@ApplicationScoped</code>	Singleton

Table 95. JAX-RS 2.0 scope and CDI scope lifecycle (continued)

Application	JAX-RS 2.0 Scope	CDI Scope annotation	Result
	Singleton	@RequestScoped	perRequest
	Singleton	@Dependent	Singleton
	Singleton	@SessionScoped	Session
	Singleton		Singleton
Provider	Singleton	@ApplicationScoped	Singleton
	Singleton	@RequestScoped	Singleton
	Singleton	@Dependent	Singleton
	Singleton	@SessionScoped	Singleton
	Singleton		Singleton

## JAX-RS 2.0 scope and CDI scope lifecycle conflict messages

The following warning messages are displayed when the scope lifecycle of JAX-RS 2.0 and CDI conflicts. They are warning messages and no actions are required.

- CWWKW1001W: The scope {1} of JAXRS-2.0 Resource {0} does not match the CDI scope {2}. Liberty gets resource instance from CDI.

This message is displayed if the JAXRS-2.0 resource scope does not match the CDI scope and the resource instance exists in CDI, so Liberty gets the resource instance from CDI. Instance does not include CDI injection if it is from JAXRS.

- CWWKW1002W: The CDI scope of JAXRS-2.0 Provider {0} is {1}. Liberty gets the provider instance from {2}.

This message is displayed because provider instance is Singleton only. Liberty gets provider instance from CDI if the CDI scope of provider is Dependent or ApplicationScoped. Instance does not include CDI injection if it is from JAXRS.

## JAX-RS 2.0 integration with managed beans

8.5.5.8

JAX-RS 2.0 in Liberty supports the use of managed beans as root resource classes, providers, and application subclasses.

- To integrate JAX-RS 2.0 with managed beans, add the `<feature>managedBeans-1.0</feature>` entry inside the `featureManager` element in the `server.xml` file.
- To use a managed bean as a JAX-RS resource, provider, or application, use the `@ManagedBean` to annotate these classes.

For example, use the Interceptors managed bean feature as follows:

```
@ManagedBean ("JaxrsManagedBean")
@Path ("/managedbean")
public class ManagedBeanResource {

 public static class MyInterceptor {
 @AroundInvoke
 public Object around(InvocationContext ctx) throws Exception {
 System.out.println("around() called");
 return ctx.proceed();
 }
 }

 @GET
 @Produces("text/plain")
 @Interceptors(MyInterceptor.class)
```

```

 public String getIt() {
 return "Hi managed bean!" ;
 }
}

```

## Restrictions on JAX-RS 2.0 with managed beans

Resource injection is only supported by the following JAX-RS component classes that are managed by Contexts and Dependency Injection (CDI):

- Application subclasses
- Providers
- Root resource classes

Specifically, to inject a managed bean instance into a certain JAX-RS component class, you must ensure that this component class can be recognized and managed as a CDI bean.

For example, to inject the `printMyName` managed bean instance into a JAX-RS root resource class as follows, you must add an empty `beans.xml` file in the `.WAR` file/`WEB-INF` repository:

```

@Path ("/managedbean")
public class ManagedBeanResource {

 @Resource(name = "printMyName")
 private PrintMyName printMyName ;

 @GET
 @Produces("text/plain")
 public String getIt() {
 printMyName .print();
 return "Hi managed bean!" ;
 }
}

@ManagedBean ("printmyname")
public class PrintMyName {

 public void print() {
 // TODO Auto-generated method stub
 System. out .println("Injection of ManagedBean is successful");
 }
}

```

## Sending multiple query parameters from Client - Cascaded or Iterated programming



8.5.5.6

You can see the following sample if you want to send multiple query parameters from the client side to the server.

### About this task

**Note:** Normally, the way to put multiple query parameters in a `WebTarget` object is by using the following cascaded programming mode:

```

javax.ws.rs.core.Response response = client.target(...).queryParam(key, value).queryParam(key, value).queryParam(key, value)

```

However, in some cases, the cascaded programming mode does not apply because the number of key value pairs is flexible and cannot be predicated. For these cases, you can use the following iteration based programming mode:

```

Map<String, String> queryStrings;
...
javax.ws.rs.client.WebTarget target = client.target(...);
for (String key: queryStrings.keySet()){
 String value = queryStrings.get(key);
 target = target.queryParam(key, value); //It is important to know queryParam method won't update current WebTarget of
}
}
javax.ws.rs.core.Response response = target.request().get();

```

## Using JAX-RS 2.0 context objects to obtain more information about requests



8.5.5.6

Java API for RESTful Web Services (JAX-RS) 2.0 provides different types of context to application subclasses, root resource classes, and providers. You can use the `@Context` annotation to inject context objects such as `HttpHeaders`, `UriInfo`, `HttpServletRequest` into class field or method parameter in application subclasses, root resource classes, and providers.

### About this task

You can use the following context objects that are available to providers (client and server), resource classes (server only), and Application subclasses (server only):

Context object	Type	Description
Application	Class	The instance of the application-supplied <code>Application</code> subclass can be injected into a class field or method parameter using the <code>@Context</code> annotation. Access to the <code>Application</code> subclass instance allows configuration information to be centralized in that class. <b>Note:</b> This <code>Application</code> subclass cannot be injected into the <code>Application</code> subclass itself since this would create a circular dependency.
UriInfo	Interface	The <code>UriInfo</code> interface provides static and dynamic, per-request information, about the components of a request URI.
HttpHeaders	Interface	The <code>HttpHeaders</code> interface provides access to request header information either in map form or via strongly typed convenience methods.
Request	Interface	The <code>Request</code> interface allows a caller to determine the best matching representation variant and to evaluate whether the current state of the resource matches any preconditions in the request.
SecurityContext	Interface	The <code>SecurityContext</code> interface provides access to information about the security context of the current request.
Providers	Interface	The <code>Providers</code> interface allows for lookup of provider instances based on a set of search criteria.

Context object	Type	Description
ResourceContext	Interface	The ResourceContext interface provides access to instantiation and initialization of resource or subresource classes in the default per-request scope.
Configuration	Interface	Both the client and the server runtime configurations are available for injection via @Context. These configurations are available for injection in providers (client or server) and resource classes (server only).

## WADL2JAVA command



8.5.5.6

The **wadl2java** command line tool processes an existing Web Application Description Language (WADL) file and generates the required artifacts for developing Java API for RESTful Web Services (JAX-RS) web service applications. The **wadl2java** command line tool supports the top-down approach to developing JAX-RS web services. When you start with an existing WADL file, use the **wadl2java** command line tool to generate the required JAX-RS artifacts.

### Web Application Description Language (WADL)

WADL is a resource-centric description language that is designed to facilitate the modeling, description, and testing of RESTful web applications. For more information, see Web Application Description Language.

### Syntax

The command syntax is as follows:

```
wadl2java --[options]
```

```
wadl2java -wadlns wadl-namespace -p package-name -sp [schema-namespace =]package-name -tMap schema-type=java-type
```

### Parameters

The following *options* values are available for the **wadl2java** command:

**-wadlns** *wadl-namespace*

Specify the WADL namespace.

**-p** *package-name*

Specifies the Java package name to use for the generated code that represents WADL resource elements.

**-sp** [*schema-namespace =*]*package-name*

Specifies the Java package name to use for the generated code that represents WADL grammar elements. Optionally specify a namespace to Java package name mapping.

**-tMap** *schema-type=java-type* \*

Specifies the optional mapping between WADL parameter or representation schema type and custom Java type.

**-repMap** *media-type=class-name* \*

Specifies the optional mapping between a WADL representation with no **wadl:element** attribute and Java class.

- resource** *resource-name*  
Specify the simple class name to use for the generated code code that represents a WADL resource without the id attribute.
- b** *binding-file-name* \*  
Specify external jaxb binding files. Use one **-b** flag for each binding file.
- catalog** *catalog-file-name*  
Specify catalog file to map the imported wadl or schema.
- d** *output-directory*  
Specify the directory into which the code is placed.
- interface**  
Specifies that interface is generated.
- impl** Specifies that a dummy service implementation is generated.
- async** *methodNames* \*  
Specifies a comma-separated list of method names or identifiers that need to support suspended asynchronous invocations.
- generateEnums**  
Specifies that Enum classes can be generated to represent parameters with multiple options.
- inheritResourceParams**  
Specifies that resource-level (path or matrix) parameters can be inherited by child resources.
- noTypes**  
Turns off generating types
- noVoidForEmptyResponses**  
Use JAX-RS Response return type for methods with no response representation.
- noAddressBinding**  
Specifies that the generator might not use the address jaxb binding file to map `wsa:EndpointReferenceType` or `wsa:EndpointReference` to `javax.xml.ws.wsaddressing.W3CEndpointReference`.
- supportMultipleXmlReps**  
Specifies that if a method contains multiple request XML representations then a separate method per every such representation is generated. Do not enable this option when a server-side JAX-RS code is generated. A single method that contains a `javax.xml.transform`. Source input parameter is generated by default such cases.
- generateResponseIfHeadersSet**  
Use JAX-RS Response return type if WADL Response element has 'header' parameters.
- generateResponseForMethods** *methodNames* \*  
Specifies a comma-separated list of method names or identifiers that need to have JAXRS Response return type generated.
- async** *methodNames* \*  
Specifies a comma-separated list of method names or identifiers that need to support suspended asynchronous invocations.
- xjc** *xjc-argumentsv* \*  
Specifies a comma-separated list of arguments that are passed directly to XJC when the JAXB data binding is used. This option causes XJC to load extra plug-ins that augment code generation. For example, to load the `toString(ts)` plug-in that adds a `toString()` method to all generated types the following *arguments* would be used: `-xjc-Xts` A list of available XJC plug-ins can be obtained by using `-xjc-X`.

**-encoding** *encoding*

Specifies the charset encoding to use when Java sources are generated.

**-h|-?|-help**

Display detailed information for options.

**-version|-v**

Display the version of the tool.

**-verbose|-V**

Specifies that the generator runs in verbose mode.

**-quiet|-q|-Q**

-quiet|-q|-Q

**wadl** wadl-url

---

## Deploying Java batch applications for Liberty



8.5.5.6

You can develop Java batch applications that are based on Java Specification Request (JSR) 352, and then submit Java batch jobs to run on a Liberty server.

### Java batch and managed batch overview



8.5.5.6

The Java batch function extends the application server to accommodate applications that must perform batch work alongside transactional applications. Batch work might take hours or even days to finish and uses large amounts of memory or processing power while it runs.

The Java Platform, Enterprise Edition Version 7 (Java EE 7) applications that are typically hosted by the product perform short, lightweight, transactional units of work. In most cases, an individual request can be completed with seconds of processor time and relatively little memory. Many applications, however, must complete batch work that is computational and resource-intensive.

Liberty supports the following batch features:

- Java batch
  - The batch-1.0 feature enables the use of the JSR-352 programming model.
- Managed batch
  - The batchManagement-1.0 feature provides the following functions:
    - A REST interface for remote job submission
    - The batchManager command-line utility
    - Job logging support
    - Multiple server support by using JMS

The batch-1.0 and the batchManagement-1.0 features support Java SE 7 and later.

**Note:** The batchManagement-1.0 feature also enables the batch-1.0 feature.

### Configuring Liberty for the batch REST API



8.5.5.6



WebSphere Application Server Liberty includes a RESTful management interface to manage your Java batch jobs. Managed batch enables a secure HTTPS REST interface so that you can externally manage your Java batch jobs.

## Procedure

1. Add the batchManagement-1.0 feature to your server.xml file.  

```
<featureManager>
 <feature>batchManagement-1.0</feature>
</featureManager>
```
2. Configure batch persistence by configuring the **databaseStore** used by the Java batch feature. Reference the **databaseStore** in the server.xml file by using the **jobStoreRef** element. The following example illustrates what your server.xml file should look like.

```
<batchPersistence jobStoreRef="BatchDatabaseStore" />
```

```
<databaseStore id="BatchDatabaseStore" dataSourceRef="batchDB" />
```

For more information on database persistence, including auto-creation versus manual creation of tables, see Java batch persistence configuration.

3. Create an SSL certificate and user registry in your server.xml file, so that batchManagement-1.0 automatically enables the SSL feature.

```
<keyStore id="defaultKeyStore" password="Liberty"/>
```

```
<basicRegistry id="basic" realm="ibm/api">
 <user name="bob" password="bobpwd" />
 <user name="jane" password="janepwd" />
</basicRegistry>
```

**Important:** The default self-signed SSL certificate in this example is intended only for development use and not for production.

For information on configuring role-based management of the batch environment and assigning users to roles, see Securing the Liberty batch environment.

## Results

The RESTful interface is now configured for the Liberty server.

## Java batch persistence configuration



8.5.5.6

Java batch uses a persistent store to persist status, checkpoints, and application persistent data across multiple runs of a job instance. The persistent store enables a job instance to be restarted if an earlier run fails or must be stopped by supplying the restarted job with the appropriate data.

### Java batch memory-based persistence configuration

The batch persistence allows a job instance to be restarted if the execution ends in a FAILED or STOPPED state. In the absence of batch persistence configuration, Java batch uses a default capability of memory-based persistence to track status, checkpoints, and application persistent data across multiple runs of a job instance.

The default memory-based persistence implementation for Java batch is used by the underlying batch container when there is no batchPersistence and databaseStore elements present in the server.xml file.

**Note:** As a limitation for Java batch memory-based persistence, by default, persistence in Java batch is based on memory. If the batch container runtime or server JVM crash or are restarted, persistence is lost. This function is intended for development purposes only and is not to be considered for production systems or critical batch processing support.

## Java batch database persistence configuration

By default, the batch run time auto-creates non-existent tables based on the server configuration defined in the **databaseStore** element. The table definitions are customized based on the **schema** and **tablePrefix** attributes of the database store.

Alternatively, the ddlGen script can be used to generate a DDL based on the server configuration. If necessary, the DDL can be customized before manually creating the tables. This DDL also incorporates server configuration such as **schema** and **tablePrefix** and contains the appropriate SQL for the database type of the data source referenced by the databaseStore.

**Note:** Customized DDL must use positive integer primary key IDs. As a limitation for database persistence, Java Batch does not accept negative or zero integer IDs persisted in the primary key identity columns. The Java Batch container runtime only runs jobs that use positive integer job IDs persisted in the primary key identity columns.

The auto-creation of tables can be disabled by using the **createTables="false"** attribute on the databaseStore. This option can be used to ensure that you use manually created tables instead of using auto-created tables if the batch runtime unexpectedly could not find your manually created tables.

The samples shown on this page use the default auto-create behavior. This behavior is equivalent to **createTables="true"**.

**Note:** To avoid data integrity issues that are caused by having an isolation level less than REPEATABLE\_READ, set the isolation level of the data source to TRANSACTION\_REPEATABLE\_READ. If you do not specify an isolation level, the default depends on the database. In most cases, the default is TRANSACTION\_REPEATABLE\_READ.

## Persistence configuration sample

The following sample configures batch access to the automatically created target database table RUNTIMEDB for Derby.

```
<!-- Batch persistence config. References a databaseStore. -->
 <batchPersistence jobStoreRef="BatchDatabaseStore" />

 <!-- The database store for the batch tables. -->
 <!-- Note this database store is referenced by the batchPersistence element. -->
 <databaseStore id="BatchDatabaseStore" dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

 <!-- Derby JDBC driver -->
 <!-- Note this library is referenced by the dataSource element -->
 <library id="DerbyLib">
 <fileset dir="{server.config.dir}/resources/derby" />
 </library>

 <!-- Data source for the batch tables. -->
 <!-- Note this data source is referenced by databaseStore element -->
 <dataSource id="batchDB" isolationLevel="TRANSACTION_REPEATABLE_READ" >
 <jdbcDriver libraryRef="DerbyLib" />
 <properties.derby.embedded
 databaseName="{server.config.dir}/resources/RUNTIMEDB"
```

```
 createDatabase="create"
 user="user"
 password="pass" />
</dataSource>
```

## Securing the Liberty batch environment



8.5.5.6

The Liberty batch framework allows you to configure role-based access to all batch management operations and also to view metadata and logs associated with your batch jobs.

### Before you begin

Three batch roles are defined by the batch container. A single user can have more than one batch role.

#### batchAdmin

A batchAdmin has unrestricted access to all batch operations. This includes permission to submit new jobs, stop and restart any user's jobs, view any job metadata and job logs that are submitted by any user in the batch domain, and to purge any jobs. A batchAdmin is not necessarily a WebSphere Application Server administrator.

#### batchSubmitter

A batchSubmitter has permission to submit new jobs and can only perform batch operations such as stop, restart, or purge on their own jobs. A batchSubmitter cannot view or modify other user's jobs. For example, if user1 and user2 are defined as batchSubmitters, and user1 submits a job, user2 cannot view the job instance data that is associated with user1's job.

#### batchMonitor

A batchMonitor has read-only permissions to all jobs. Users in this role can view all job instances and executions and have access to all job logs. A batchMonitor cannot submit their own jobs, or stop, restart, or purge any jobs.

**Note:** Once batch security is enabled, any JobOperator API method or REST operation that returns a list will be filtered based on the batch roles granted to the current user. For example, when a user with only batchSubmitter permissions requests a list of all job instances, only job instances submitted by the current user will be returned.

### About this task

#### Procedure

1. By default, the batch-1.0 feature does not enable any security. The JobOperator methods are left open for all users whether they are authenticated or not. The methods are left open for development purposes only and require no security configuration. The batchManagement-1.0 feature enables the batch REST API. The REST API always requires a user to authenticate even when the appSecurity-2.0 feature is not enabled, but all users will be treated as batch administrators and can perform all batch operations on any job instance. Once appSecurity-2.0 is enabled, batch role-based security authorization will be performed and users will be restricted to batch operations defined by their given batch roles.
  - a. Enable the batch role-based security through the JobOperator API

```
<featureManager>
 <feature>batch-1.0</feature>
 <feature>appSecurity-2.0</feature>
</featureManager>
```
  - b. Enable the batch role-based security through the REST API.

**Note:** The batchManagement-1.0 feature includes the batch-1.0 feature.

```

 <featureManager>
 <feature>batchManagement-1.0</feature>
 <feature>appSecurity-2.0</feature>
 </featureManager>

```

2. Configure your server.xml file to support role-based security. The following example illustrates a basic user registry that defines a list of users. This registry is used by sample batch role based security configurations.

```

<basicRegistry id="basic" realm="ibm/api">
 <user name="alice" password="alicepwd" />
 <user name="bob" password="bobpwd" />
 <user name="jane" password="janepwd" />
 <user name="joe" password="joepwd" />
 <user name="phyllis" password="phyllispwd" />
 <user name="kai" password="kaipwd" />
</basicRegistry>

```

In this example, one user occupies multiple roles.

```

<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
 <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
 <user name="jane" />
 <user name="phyllis" />
 <user name="bob" />
 </security-role>
 <security-role name="batchMonitor" >
 <user name="joe" />
 <user name="bob" />
 </security-role>
</authorization-roles>

```

In this example, one user occupies multiple roles. All users have the batchSubmitter role.

```

<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
 <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
 <special-subject type="ALL_AUTHENTICATED_USERS"/>
 </security-role>
 <security-role name="batchMonitor" >
 <user name="joe" />
 <user name="bob" />
 </security-role>
</authorization-roles>

```

In this example, one user occupies multiple roles. All users, including those users that are not authenticated, are allowed to have the batchMonitor role.

```

<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
 <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
 <user name="joe" />
 <user name="bob" />
 </security-role>
 <security-role name="batchMonitor" >
 <special-subject type="EVERYONE"/>
 </security-role>
</authorization-roles>

```

## Java batch shutdown and recovery



8.5.5.6

The Java batch feature behaves differently when the server is shut down while jobs are still running.

### Deactivating Java batch

Java batch is deactivated when the server is stopped or the Java batch feature is removed. It is also deactivated and reactivated to process a dynamic configuration update. Deactivating Java batch causes stop requests to be issued to all active jobs and messages to be logged for each stop. There is a 2-second period for the jobs to stop. If after 2 seconds there are Java batch jobs that are still running, a message is logged that indicates which job execution IDs are still active. Java batch then shuts down. If jobs are still running after Java batch shuts down, they can experience unpredictable behavior.

**Important:** If the server is unexpectedly stopped, Java batch recovers by marking all jobs that were running on the server but not completed as FAILED when the server is restarted.

### Batch REST API



8.5.5.6

The WebSphere Application Server Liberty includes a RESTful management interface to manage your batch jobs.

The basic operations that are associated with a batch job are to submit (start), stop, restart, and view status. You can perform these operations by using any HTTP REST client. Any data that is submitted as part of a request or returned as part of a response is JSON formatted.

The following examples show the functions that you can perform with the REST API.

**Note:** The Batch REST API is versioned on a URL by URL basis. URL's with no version number are considered as version 1 of the API. The web addresses of the batch REST interface all start with the root web address: `https://{host}:{port}/ibm/api/batch/{version}/`.

**Remember:** The web addresses of the batch REST interface all start with the root web address: `https://{host}:{port}/ibm/api/batch`.

- “REST API with a user ID that is only a submitter”
- “Job instances” on page 1374
- “Job executions” on page 1381
- “Step executions” on page 1382
- “Job logs” on page 1385
- “HTTP Return Codes” on page 1386
- “STOP requests in a distributed server batch environment” on page 1386
- “JOBLOGS requests in a distributed server batch environment” on page 1386
- “Purge requests in a distributed server batch environment” on page 1387

### REST API with a user ID that is only a submitter

The results returned by the GET (“read”) URLs will be filtered when the user ID that invokes the REST API over HTTPS has only been granted access to the **batchSubmitter** role. The submitter only sees instance and execution data that is associated with job instances that were submitted by the submitter himself. In contrast, user ids granted access to the **batchAdmin** and **batchMonitor** roles will be able to view all instance and execution data (returned by any given URL with any given set of parameters).

A userid which only has access to the **batchSubmitter** role sees results that are first filtered by the standard parameters as described in the documentation of the given URL, and then further filtered by only returning instance and execution data associated with job instances submitted by the submitter himself.

See *Securing the Liberty batch environment* for more information.

## Job instances

### GET /ibm/api/batch/jobinstances/

This URI returns a list of job instances. Query parameters include:

- `page=[page number]`: Indicates which page (subset of records) to return. The default is 0.
- `pageSize=[number of records per page]`: Indicates the number of records per page. The default is 50.

Sample requests:

```
https://localhost:9443/ibm/api/batch/jobinstances
```

```
https://localhost:9443/ibm/api/batch/jobinstances?page=13&pagesize=20
```

Sample response:

```
[
 {
 "jobName": "test_sleepyBatchlet",
 "instanceId": 7,
 "appName": "SimpleBatchJob#SimpleBatchJob.war",
 "submitter": "bob",
 "batchStatus": "COMPLETED",
 "jobXMLName": "test_sleepyBatchlet",
 "instanceState": "COMPLETED",
 "_links": [
 {
 "rel": "self",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/7"
 },
 {
 "rel": "job logs",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/7/joblogs"
 }
]
 },
 {
 "jobName": "test_sleepyBatchlet",
 "instanceId": 6,
 "appName": "SimpleBatchJob#SimpleBatchJob.war",
 "submitter": "bob",
 "batchStatus": "COMPLETED",
 "jobXMLName": "test_sleepyBatchlet",
 "instanceState": "COMPLETED",
 "_links": [
 {
 "rel": "self",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/6"
 },
 {
 "rel": "job logs",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/6/joblogs"
 }
]
 }
]
```

### GET /ibm/api/batch/v2/jobinstances/

This URI returns a list of job instances filtered by the following query parameters:

- `jobInstanceId=[instanceId]:[instanceId]`: Returns job instances equal to and between the `instanceId` range.
- `jobInstanceId=>[instanceId]`: Returns job instances equal to and greater than the provided `instanceId`.
- `jobInstanceId=<[instanceId]`: Returns job instances equal to and less than the provided `instanceId`.
- `jobinstanceId=[instanceId],[instanceId],[instanceId]`: Returns job instances specified.
- `createTime=[yyyy-MM-dd]:[yyy-MM-dd]`: Returns job instances between the date range inclusively. For more information, see note further on in this topic.
- `createTime=[yyyy-MM-dd]`: Returns job instances on the given date. For more information, see note further on in this topic.
- `createTime=>3d`: Returns job instances that were created on or after the day which was three days ago. For example, the `createTime` is greater than or equal to the beginning of the day three days ago. For more information, see note further on in this topic.
- `createTime=<3d`: Returns job instances that were created on or before the day which was three days ago. For example, the `createTime` is less than or equal to the end of the day three days ago. For more information, see note further on in this topic.
- `instanceState=[state],[state]`: Returns job instances with the provided state. Valid Instance States are SUBMITTED, JMS\_QUEUED, JMS\_CONSUMED, DISPATCHED, FAILED, STOPPED, COMPLETED and ABANDONED.
- `exitStatus=[string]`: Returns job instances matching the exit status string. The string criteria may utilize the wildcard(\*) operator on either end.
- `page=[page number]`: Indicates which page (subset of records) to return. The default is 0.
- `pageSize=[number of records per page]`: Indicates the number of records per page. The default is 50.

**Note: The role of the server default timezone in queries involving createTime**

When you submit a job, the `createTime` for the job instance is stored in the job repository and normalized to UTC. When specifying dates via `yyyy-MM-dd`, as in `createTime=[yyyy-MM-dd]` or `createTime=[yyyy-MM-dd]:[yyy-MM-dd]:`, then you must convert the `yyyy-MM-dd` date string into a specific range of UTC times to match against the `createTime` values in the job instance table records. To do this, the application uses the default timezone of the server handling the REST request. It is this server's timezone that is used to convert the date string into a UTC time range that will be matched against.

The following table illustrates data returned by the query parameters **jobInstanceId=10:13**.

JOBINSTANCEID	CREATETIME(* in server1's timezone)	INSTANCESTATE	EXITSTATUS
10	11-05-2015.01:10:00	COMPLETED	SUCCESS
11	11-08-2014.02:20:00	COMPLETED	SUCCESS
12	11-10-2015.03:30:00	FAILED	FAILURE
13	11-11-2015.04:40:00	COMPLETED	SUCCESS

\*Since the job repository stores the `createTime` in a UTC format, it is important to understand that the table data above shows the `createTime` formatted using the server's default timezone of a particular server, for example, "server1". If we had constructed a similar table from the perspective of a second server with a different default timezone (than "server1"), we would show a different set of `CREATETIME` column values which a corresponding timezone-based difference. It is the default timezone of the server handling the REST request which is used to map `yyyy-MM-dd` date string parameters to UTC `createTime` values in the database in the examples below.

The following commands return the same result regardless of which server they are issued against:

- `jobInstanceId=11:13` would return JOBINSTANCEID's 11, 12 and 13.
- `jobInstanceId=>12` would return JOBINSTANCEID's 12 and 13.
- `jobInstanceId=<12` would return JOBINSTANCEID's 11 and 12.
- `jobInstanceId=11,12` would return JOBINSTANCEID's 11 and 12.
- `instanceState=FAILED` would return JOBINSTANCEID 12..
- `exitStatus=SUCCESS` would return JOBINSTANCEID's 10, 11 and 13.

The following commands may return different results against servers with different default timezones. In these examples, they are issued against "server1" (the same server used to populate the table above) at date and time: 11-11-2015:07:00:00 in the server's default timezone.

- `createTime=>2d` would return JOBINSTANCEID's 12 and 13 (on or after 2 days ago, which was 11-09-2015)
- `createTime=<2d` would return JOBINSTANCEID's 10 and 11 (on or before 2 days ago, which was 11-09-2015)
- `createTime=2015-11-08:2015-11-11` would return records with JOBINSTANCEID's 11,12, and 13.
- `createTime=2015-11-10` would return record with JOBINSTANCEID 12.

Sample requests:

```
https://localhost:9443/ibm/api/batch/v2/jobinstances?instanceId=20:50
```

```
https://localhost:9443/ibm/api/batch/jobinstances?createTime=>2d
```

```
https://localhost:9443/ibm/api/batch/v2/jobinstances?instanceId=4,12,17&instanceState=COMPLETED
```

```
https://localhost:9443/ibm/api/batch/v2/jobinstances?instanceId=4500:4600&createTime=2015-11-27&instanceState=COMPLETED
```

#### GET /ibm/api/batch/jobinstances/job *instance id*

This URI returns detailed information about the specified job instance such as all executions that are associated with a specified job instance. Results are returned in order from most recent to the oldest. The most recent result is displayed first in the list.

Sample response:

```
{
 "jobName": "test_sleepyBatchlet",
 "instanceId": 7,
 "appName": "SimpleBatchJob#SimpleBatchJob.war",
 "submitter": "bob",
 "batchStatus": "COMPLETED",
 "jobXMLName": "test_sleepyBatchlet",
 "instanceState": "COMPLETED",
 "_links": [
 {
 "rel": "self",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/7"
 },
 {
 "rel": "job logs",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/7/joblogs"
 },
 {
 "rel": "job execution",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/7/jobexecutions/7"
 }
]
}
```

#### POST /ibm/api/batch/jobinstances/

Use this URI to submit (start) a new job.

The following example illustrates the request body for submitting a job that is packaged in a WAR module, in JSON formatting:



```
{
 "applicationName" : "SimpleBatchJob",
 "moduleName" : "SimpleBatchJob.war",
 "jobXMLName" : "test_batchlet_jsl",
 "jobParameters" : { "prop1" : "prop1value", "prop2" : "prop2value"}
}
```

The following example illustrates the request body for submitting a job that is packaged in an EJB module, in JSON formatting:

```
{
 "applicationName" : "SimpleBatchJob",
 "moduleName" : "SimpleBatchJobEJB.jar",
 "componentName" : "MyEJB",
 "jobXMLName" : "test_batchlet_jsl",
 "jobParameters" : { "prop1" : "prop1value", "prop2" : "prop2value"}
}
```

The *applicationName* identifies the batch application. It is required unless *moduleName* is specified, in which case the *applicationName* is derived from the *moduleName* by trimming off the .war or .jar suffix of the *moduleName*. For example, if you provide no *applicationName* and *moduleName*=SimpleBatchJob.war, then *applicationName* defaults to SimpleBatchJob.

The *moduleName* identifies the module within the batch application that contains the job artifacts, such as the JSL. The job is submitted under the module's component context. The *moduleName* is required unless *applicationName* is specified, in which case the *moduleName* is derived from the *applicationName* by appending .war to the *applicationName*. For example, if you provide *applicationName*=SimpleBatchJob and no *moduleName*, then *moduleName* defaults to SimpleBatchJob.war.

The *componentName* identifies the EJB component within the batch application EJB module. If specified, the job is submitted under the EJB's component context.

**Note:** The *componentName* is required only when the module is an EJB module. When the module is a WAR module, the *componentName* is not required.

You must enter a value for *jobXMLName*. The value for *jobParameters* is optional.

**8.5.5.7** As an alternative to using the JSL job definition that is packaged within your batch application under META-INF/batch-jobs, you can pass your JSL inline as part of your REST job submission request. The JSL that is submitted inline always overrides any JSL that is packaged with the batch application. There are two ways to submit your JSL inline as part of your HTTP request.

1. Include your JSL as a JSON property in your job submission request. Add the property **jobXML** to the JSON object and add the entire content of the JSL file as a JSON string as the value of the property.

**Note:** You must properly escape the XML string so it can be parsed by a JSON parser. Most JSON libraries will do this for you.

The following example illustrates the request body for submitting a job that utilizes a single part HTTP request with the JSL being passed in-line by the JSON.

```
{
 "applicationName":"SimpleBatchJob",
 "jobXMLName":"test_multiPartition_3Steps",
 "jobXML":"<?xml version='1.0' encoding='UTF-8'>
standalone='yes'?>
\r\n<job id='\"test_multiPartition_3Steps\"'
xmlns='\"http://xmlns.jcp.org/xml/ns/javaee\"'>
\r\n\t<step id='\"step1\"' next='\"step2\"'>
\r\n\t\t<batchlet ref='\"simpleBatchlet\"'/>
\r\n\t\t<partition>\r\n\t\t\t<plan partitions='\"3\"'/>
\r\n\t\t\t</partition>\r\n\t\t</step>
\r\n\t<step id='\"step2\"' next='\"step3\"'>
```

```

\r\n\t\t<batchlet ref="simpleBatchlet" />
\r\n\t\t<partition>\r\n\t\t\t<plan partitions="3" />
\r\n\t\t</partition>\r\n\t\t</step>\r\n\t\t<step id="step3">
\r\n\t\t\t<batchlet ref="simpleBatchlet" />\r\n\t\t</partition>
\r\n\t\t\t<plan partitions="3"/>
\r\n\t\t</partition>\r\n\t\t</step>\r\n</job>
}

```

**Note:** The **jobXML** field must be parsed by a JSON parser and marshalled into a valid JSON object. The **jobXMLName** field is optional, as the job ID information in the inline JSL is used for the job name.

2. Use an HTTP multi-part form. When you use an HTTP multipart form the JSON job submission data and XML job definition need to be submitted as two separate parts of the HTTP body. The JSON part of the multi-part form must be named *jobdata* and the XML part of the form must be named *jsl*. The XML does not need to be marshalled to a JSON string when you use a multi-part form.

The following example illustrates the HTTP request for submitting a job that uses a multi-part HTTP request with the JSL being passed inline via the *jsl* message part.

```
Content-Type: multipart/form-data;boundary=-----49424d5f4a4241544348
```

```
-----49424d5f4a4241544348
```

```
Content-Disposition: form-data; name="jobdata"
```

```
Content-Type: application/json; charset=UTF-8
```

```
{
 "applicationName" : "SimpleBatchJob",
 "moduleName" : "SimpleBatchJob.war",
 "jobXMLName" : "test_multiPartition"
}
```

```
-----49424d5f4a4241544348
```

```
Content-Disposition: form-data; name="jsl"
```

```
Content-Type: application/xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<job id="test_multiPartition" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
<step id="step1">
<batchlet ref="simpleBatchlet" />
<partition>
 <plan partitions="3" />
</partition>
</step>
</job>
```

```
-----49424d5f4a4241544348--
```

**Note:** The **jobXMLName** field is optional, as the job ID information in the inline JSL is used for the job name.

The following sample response illustrates a successful job submission:

```
{
 "jobName": "test_sleepyBatchlet",
 "instanceId": 10,
 "appName": "SimpleBatchJob#SimpleBatchJob.war",
 "submitter": "bob",
 "batchStatus": "STARTING",
 "jobXMLName": "test_sleepyBatchlet",
 "instanceState": "SUBMITTED",
 "_links": [
 {
 "rel": "self",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/10"
 }
],
}
```

```

 {
 "rel": "job logs",
 "href": "https://localhost:9443/ibm/api/batch/jobinstances/10/joblogs"
 }
]
}

```

**PUT /ibm/api/batch/jobinstances/job *instance id*?action=stop**

Use this URI to stop the most recent job execution that is associated with this job instance if it is running. If it is not, the API returns an error.

**PUT /ibm/api/batch/jobinstances/job *instance id*?action=restart**

Use this URI to restart the most recent job execution that is associated with this job instance only if it is in STOPPED or FAILED state. If no job execution is associated with this instance, or the latest job execution is in COMPLETED state, the API returns an error.

**PUT /ibm/api/batch/jobinstances/job *instance id*?action=restart&reusePreviousParams=true**

Use this URI to restart the most recent job execution and reuse the job parameters from the previous execution that is associated with this job instance. The previous execution must be in STOPPED or FAILED state. If no job execution is associated with this instance, or the latest job execution is in COMPLETED state, then the API returns an error. Note that **reusePreviousParams** is an optional setting. The default setting is **reusePreviousParams=false**.

**Note:** When **reusePreviousParams=true**, any job parameters that are submitted as part of the current restart request take precedence over any previous job parameters. Current parameters override previous parameters with the same job parameter key name.

**DELETE /ibm/api/batch/jobinstances/job *instance id***

This URI purges all database entries and job logs that are associated with this job instance. This API returns an error if the job instance has active job executions. If there is an error when you delete the job logs, then no attempt is made to delete the job instance data from the job store database. Query parameters include:

- `purgeJobStoreOnly=true|false`: When `purgeJobStoreOnly=true`, no attempt is made to purge the job logs associated with this job instance. The default setting is `purgeJobStoreOnly=false`. This API returns an error if the job instance has active job executions.

**Note:** No purge response message is returned when you utilize the single purge API.

**DELETE /ibm/api/batch/v2/jobinstances/**

This URI purges all database entries and job logs associated with the job instances returned by the following purge filter parameters:

**Note:** It is recommended that you utilize the GET interface to list out the jobs and verify they are the correct jobs to purge before performing the DELETE interface to purge them.

- `page=[page number]`: Indicates which page (subset of records) to return. The default is 0
- `pageSize=[number of records per page]`: Indicates the number of records per page. The default is 50.
- `purgeJobStoreOnly=true|false`: When `purgeJobStoreOnly=true`, no attempt is made to purge the job logs associated with this job instance. The default setting is `purgeJobStoreOnly=false`. This API returns an error if the job instance has active job executions.
- `jobInstanceId=[instanceId]:[instanceId]`: Purges job instances equal to and between the `instanceId` range.
- `jobInstanceId=>[instanceId]`: Purges job instances equal to and greater than the provided `instanceId`.
- `jobInstanceId=<[instanceId]`: Purges job instances equal to and less than the provided `instanceId`.
- `jobinstanceId=[instanceId],[instanceId],[instanceId]`: Purges job instances specified.

- createTime=[yyyy-MM-dd]:[yyy-MM-dd]: Returns job instances between the date range inclusively. For more information, see note further on in this topic.
- createTime=[yyyy-MM-dd]: Returns job instances on the given date. For more information, see note further on in this topic.
- createTime=>3d: Returns job instances that were created on or after the day which was three days ago. For example, the createTime is greater than or equal to the beginning of the day three days ago. For more information, see note further on in this topic.
- createTime=<3d: Returns job instances that were created on or before the day which was three days ago. For example, the createTime is less than or equal to the end of the day three days ago. For more information, see note further on in this topic.
- instanceState=[state],[state]: Purges job instances with the provided state. Valid Instance States are SUBMITTED, JMS\_QUEUED, JMS\_CONSUMED, DISPATCHED, FAILED, STOPPED, COMPLETED, and ABANDONED.
- exitStatus=[string]: Returns job instances matching the exit status string. The string criteria may utilize the wildcard(\*) operator on either end.

**Note:** By default, unless the page and pageSize parameters are specified, a maximum of 50 records are purged.

**Note: The role of the server default timezone in queries involving createTime**

When you submit a job, the createTime for the job instance is stored in the job repository and normalized to UTC. When specifying dates via yyyy-MM-dd, as in createTime=[yyyy-MM-dd] or createTime=[yyyy-MM-dd]:[yyy-MM-dd]:, then you must convert the yyyy-MM-dd date string into a specific range of UTC times to match against the createTime values in the job instance table records. To do this, the application uses the default timezone of the server handling the REST request. It is this server's timezone that is used to convert the date string into a UTC time range that will be matched against.

The following table illustrates data returned by the query parameters **jobInstanceId=10:13**.

JOBINSTANCEID	CREATETIME(* in server1's timezone)	INSTANCESTATE	EXITSTATUS
10	11-05-2015.01:10:00	COMPLETED	SUCCESS
11	11-08-2014.02:20:00	COMPLETED	SUCCESS
12	11-10-2015.03:30:00	FAILED	FAILURE
13	11-11-2015.04:40:00	COMPLETED	SUCCESS

\*Since the job repository stores the createTime in a UTC format, it is important to understand that the table data above shows the createTime formatted using the server's default timezone of a particular server, for example, "server1". If we had constructed a similar table from the perspective of a second server with a different default timezone (than "server1"), we would show a different set of CREATETIME column values which a corresponding timezone-based difference. It is the default timezone of the server handling the REST request which is used to map yyyy-MM-dd date string parameters to UTC createTime values in the database in the examples below.

The following commands return the same result regardless of which server they are issued against:

- jobInstanceId=11:13 would return JOBINSTANCEID's 11, 12 and 13.
- jobInstanceId=>12 would return JOBINSTANCEID's 12 and 13.
- jobInstanceId=<12 would return JOBINSTANCEID's 11 and 12.
- jobInstanceId=11,12 would return JOBINSTANCEID's 11 and 12.
- instanceState=FAILED would return JOBINSTANCEID 12..

- `exitStatus=SUCCESS` would return `JOBINSTANCEID`'s 10, 11 and 13.

The following commands may return different results against servers with different default timezones. In these examples, they are issued against "server1" (the same server used to populate the table above) at date and time: 11-11-2015:07:00:00 in the server's default timezone.

- `createTime=>2d` would return `JOBINSTANCEID`'s 12 and 13 (on or after 2 days ago, which was 11-09-2015)
- `createTime=<2d` would return `JOBINSTANCEID`'s 10 and 11 (on or before 2 days ago, which was 11-09-2015)
- `createTime=2015-11-08:2015-11-11` would return records with `JOBINSTANCEID`'s 11,12, and 13.
- `createTime=2015-11-10` would return record with `JOBINSTANCEID` 12.

Sample response:

```
[{"instanceId":394,"purgeStatus":"COMPLETED","message":"Successful purge.,"redirectUrl":""},
{"instanceId":395,"purgeStatus":"COMPLETED","message":"Successful purge.,"redirectUrl":""},
{"instanceId":396,"purgeStatus":"COMPLETED","message":"Successful purge.,"redirectUrl":""},
{"instanceId":397,"purgeStatus":"COMPLETED","message":"Successful purge.,"redirectUrl":""},
{"instanceId":398,"purgeStatus":"COMPLETED","message":"Successful purge.,"redirectUrl":""}]
```

The following `purgeStatus` values can be returned:

#### **COMPLETED**

Indicates that the job purge completed successfully.

#### **FAILED**

Indicates that the job purge failed.

#### **STILL\_ACTIVE**

Indicates that the job purge failed because it was still active.

#### **JOBLOGS\_ONLY**

Indicates that the database purge failed, but that the job logs were successfully purged.

#### **NOT\_LOCAL**

Indicates that the job purge failed because the job is not local.

## **Job executions**

### **GET /ibm/api/batch/jobexecutions/job execution id**

This URI returns detailed information about a specified job execution and includes links to associated step executions and job logs.

Sample request:

```
https://localhost:9443/ibm/api/batch/jobexecutions/9
```

Sample response:

```
{
 "jobName":"test_sleepyBatchlet",
 "executionId":9,
 "instanceId":9,
 "batchStatus":"COMPLETED",
 "exitStatus":"COMPLETED",
 "createTime":"2015/05/07 16:09:41.025 -0400",
 "endTime":"2015/05/07 16:09:52.127 -0400",
 "lastUpdatedTime":"2015/05/07 16:09:52.127 -0400",
 "startTime":"2015/05/07 16:09:41.327 -0400",
 "jobParameters":{
 },
 "restUrl":"https://localhost:9443/ibm/api/batch",
 "serverId":"localhost/C:/ibm/RAD_workspaces/Liberty7/build.image/wlp/usr/server1",
 "logpath":"C:\\ibm\\Liberty\\wlp\\usr\\servers\\server1\\logs\\joblogs\\test_sleepyBatchlet\\2015-05-07\\ins",
 "stepExecutions":[
]
}
```

```

 "stepExecutionId":9,
 "stepName":"step1",
 "batchStatus":"COMPLETED",
 "exitStatus":"SleepyBatchlet:i=10;stopRequested=false",
 "stepExecution":"https://localhost:9443/ibm/api/batch/jobexecutions/9/stepexecutions/step1"
 },
 "_links":[
 {
 "rel":"self",
 "href":"https://localhost:9443/ibm/api/batch/jobexecutions/9"
 },
 {
 "rel":"job instance",
 "href":"https://localhost:9443/ibm/api/batch/jobinstances/9"
 },
 {
 "rel":"step executions",
 "href":"https://localhost:9443/ibm/api/batch/jobexecutions/9/stepexecutions"
 },
 {
 "rel":"job logs",
 "href":"https://localhost:9443/ibm/api/batch/jobexecutions/9/joblogs"
 },
 {
 "rel":"stop url",
 "href":"https://localhost:9443/ibm/api/batch/jobexecutions/9?action=stop"
 }
]
}

```

#### GET /ibm/api/batch/jobexecutions/job execution id/jobinstance

This URI returns detailed information about the job instance related to the specified job execution.

#### GET /ibm/api/batch/jobinstances/job instance id/jobexecutions

This URI returns detailed information about the job execution(s) for a specified job instance. This includes links to associated step executions and job logs.

#### GET /ibm/api/batch/jobinstances/job instance id/jobexecutions/job execution sequence number

This URI returns detailed information about the specified job execution in relation to the specified job instance ID. This includes links to associated step executions and job logs.

**Note:** The job execution sequence number means the 0th, 1st, 2nd, etc. job execution related to the specified job instance.

#### PUT /ibm/api/batch/jobexecutions/job execution id?action=stop

Use this URI to stop the specified running job execution. Required parameters include action = *stop*, *restart*.

#### PUT /ibm/api/batch/jobexecutions/job execution id?action=restart

Use this URI to restart the specified job execution. Required parameters include action = *stop*, *restart*.

## Step executions

#### GET /ibm/api/batch/jobexecutions/job execution id/stepexecutions

This URI returns a JSON array of all the step execution details for the specified job execution. If your job contains a partitioned step, the partition information will be returned listed within each step.

Sample request:

```
https://localhost:8020/ibm/api/batch/jobexecutions/40/stepexecutions
```

The following sample illustrates a response for a partitioned step.

```

[
 {
 "stepExecutionId":47,
 "executionId":39,
 "instanceId":35,
 "stepName":"step1",
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:08.652 -0400",
 "endTime":"2015/03/30 11:10:09.817 -0400",
 "exitStatus":"COMPLETED",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 },
 "partitions":[
 {
 "partitionNumber":0,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:09.579 -0400",
 "endTime":"2015/03/30 11:10:09.706 -0400",
 "exitStatus":"step1",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 },
 {
 "partitionNumber":1,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:09.257 -0400",
 "endTime":"2015/03/30 11:10:09.302 -0400",
 "exitStatus":"step1",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 },
 {
 "partitionNumber":2,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:09.469 -0400",
 "endTime":"2015/03/30 11:10:09.548 -0400",
 "exitStatus":"step1",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 }
]
 }
]

```

```

 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
}
],
{
 "stepExecutionId":51,
 "executionId":39,
 "instanceId":35,
 "stepName":"step2",
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:09.915 -0400",
 "endTime":"2015/03/30 11:10:10.648 -0400",
 "exitStatus":"COMPLETED",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 },
 "partitions":[
 {
 "partitionNumber":0,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:10.324 -0400",
 "endTime":"2015/03/30 11:10:10.417 -0400",
 "exitStatus":"step2",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 },
 {
 "partitionNumber":1,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:10.260 -0400",
 "endTime":"2015/03/30 11:10:10.347 -0400",
 "exitStatus":"step2",
 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 },
 {
 "partitionNumber":2,
 "batchStatus":"COMPLETED",
 "startTime":"2015/03/30 11:10:10.507 -0400",
 "endTime":"2015/03/30 11:10:10.557 -0400",
 "exitStatus":"step2",

```



```

 "metrics":{
 "READ_COUNT":"0",
 "WRITE_COUNT":"0",
 "COMMIT_COUNT":"0",
 "ROLLBACK_COUNT":"0",
 "READ_SKIP_COUNT":"0",
 "PROCESS_SKIP_COUNT":"0",
 "FILTER_COUNT":"0",
 "WRITE_SKIP_COUNT":"0"
 }
 },
 {
 "_links":[
 {
 "rel":"job execution",
 "href":"https://localhost:9443/ibm/api/batch/jobexecutions/9"
 },
 {
 "rel":"job instance",
 "href":"https://localhost:9443/ibm/api/batch/jobinstances/9"
 }
]
 }
]

```

#### GET /ibm/api/batch/jobexecutions/job execution id/stepexecutions/step name

This URI returns a JSON array containing the step execution details for the specified job execution and step name.

#### GET /ibm/api/batch/jobinstances/job instance id/jobexecutions/job execution sequence number/stepexecutions/step name

This URI returns a JSON array containing the step execution details for the specified job instance, job execution, and step name.

#### GET /ibm/api/batch/stepexecutions/step execution id

This URI returns a JSON array containing the step execution details for the specified step execution.

## Job logs

#### GET /ibm/api/batch/jobinstances/job instance id/joblogs

This URI returns a JSON array with REST links to all job log parts for the specified job instance.

#### GET /ibm/api/batch/jobexecutions/job execution id/joblogs

This URI returns a JSON array with REST links to all job log parts for the specified job execution.

**Important:** The following example shows the format of the REST links.

If your job execution has the following job log parts,

```

joblogs/instance.inst-id/execution.exec-id/part.1.log
joblogs/instance.inst-id/execution.exec-id/part.2.log
joblogs/instance.inst-id/execution.exec-id/step.step-name/partition.0/part.1.log
joblogs/instance.inst-id/execution.exec-id/step.step-name/partition.1/part.1.log

```

then the corresponding REST links are:

```

/ibm/api/batch/jobexecutionsexec-id/joblogs?part=part.1.log
/ibm/api/batch/jobexecutionsexec-id/joblogs?part=part.2.log
/ibm/api/batch/jobexecutionsexec-id/joblogs?part=step.step-name/partition.0/part.1.log
/ibm/api/batch/jobexecutionsexec-id/joblogs?part=step.step-name/partition.1/part.1.log

```

Optional parameters include:

**type = text**

Text returns all job logs as plain text. All job log parts are aggregated together. Part delimiting header and footer records are inserted into the stream to delimit the different parts as they are aggregated together.

**type = zip**

Zip returns all job logs for the specified job instance or job execution as a compressed file. The directory structure of the job logs is preserved in the compressed file.

**GET /ibm/api/batch/jobinstances/job instance id/joblogs?type=text | zip**

**GET /ibm/api/batch/jobexecutions/job execution id/joblogs?type=text | zip**

The behavior of these two URIs with the **type** parameter specified varies by value.

**type = text**

Text returns all job logs as plain text. All job log parts are aggregated together. Part delimiting header and footer records are inserted into the stream to delimit the different parts as they are aggregated together.

**type = zip**

Zip returns all job logs for the specified job instance or job execution as a compressed file. The directory structure of the job logs is preserved in the compressed file.

**GET /ibm/api/batch/jobexecutions/job execution id/joblogs?part=path to part&type=text | zip**

With the **part** parameter specified, this URI returns job log parts as either plain text (type=text) or in a compressed file (type=zip). The default setting is type=text.

## HTTP Return Codes

The following HTTP return codes for the REST API.

- HTTP 200 OK
- HTTP 201 Successfully created a new resource.
- HTTP 202 Accepted request, but processing is not complete.
- HTTP 400 Bad Request with invalid parameters. See returned message for details.
- HTTP 401 Unauthorized to access this resource.
- HTTP 403 Authentication failed.
- HTTP 404 The requested resource cannot be found or does not exist.
- HTTP 409 The request conflicts with the current state of the resource. See returned message for details.
- HTTP 500 Internal Server Error.

## STOP requests in a distributed server batch environment

Stop requests sent to the batch REST API must be sent directly to the executor where the job is running. If a stop request is sent to a dispatcher or executor where the job is not running, the request is redirected to the correct executor by an HTTP 302 redirection response message. The **location** field in the HTTP 302 redirection response indicates the correct URL to use for the stop request.

## JOBLOGS requests in a distributed server batch environment

Job logs requests sent to the batch REST API must be sent directly to the executor where the job is running. If a job logs request is sent to a dispatcher or executor where the job is not running, the request is redirected to the correct executor by an HTTP 302 redirection response message. The **location** field in the HTTP 302 redirection response indicates the correct URL to use for the job logs request.

**Note:** Job logs requests that are sent to the batch REST API for an entire job instance work only if all job executions for that instance ran on the same executor. If the executions ran on different executors, then the job logs requests for the instance fails. In this case, you must fetch the job logs for each execution separately.

## Purge requests in a distributed server batch environment

Purge requests sent to the batch REST API must be sent directly to the executor where the job is running. If a purge request is sent to a dispatcher or executor where the job is not running, the request is redirected to the correct executor by an HTTP 302 redirection response message. The **location** field in the HTTP 302 redirection response indicates the correct URL to use for the purge request.

**Note:** Purge requests that are sent to the batch REST API for an entire job instance work only if all job executions for that instance ran on the same executor. If the executions ran on different executors, then the purge requests for the instance fails.

## Enabling multiple server support by using the Liberty embedded messaging provider



8.5.5.6

The batch environment can be set up to have servers function as batch dispatchers while other servers function as batch executors. Batch dispatchers accept requests from external clients and make them available to the batch executors. The batch executors receive requests that match its defined capabilities and execute those requests. Batch dispatchers and batch executors communicate by using Java Messaging Service (JMS).

### Before you begin

Determine where the embedded message engine is hosted. It can be hosted on the batch dispatch server, the batch executor server, or on a separate server. This server must be configured before you complete this task. The JMS connection factory and activation specification references the message engine server in its configuration. To configure the message engine:

1. Add the **wasJmsServer-1.0** feature to the `server.xml`.
2. Define the message engine by adding the **messageEngine** element. Define the queue that is used for the batch dispatcher and batch executor. The following example illustrates the message engine configuration in your `server.xml`.

```
<!--specify the ports for the message engine.
The ports in this example are the default ports.
This element is not needed when the default ports are used. -->
<wasJmsEndpoint host="*"
 wasJmsPort="7280"
 wasJmsSSLPort="7290"
 enabled="true">
</wasJmsEndpoint>

<messagingEngine>
 <!-- queue for batch jms message. -->
 <queue id="batchLibertyQueue"
 forceReliability="ReliablePersistent"
 receiveAllowed="true"/>
</messagingEngine>
```

### About this task

This task helps you configure the batch dispatch server and the batch executor by using the Liberty embedded messaging provider.

## Procedure

To configure batch dispatcher and executor that uses Liberty embedded messaging provider:

1. Configure the batch JMS dispatcher.
  - a. Enable JMS support by adding the **wasJmsClient-2.0** feature to the feature manager in your `server.xml`.
  - b. Add the **batchJmsDispatcher** element to your `server.xml` on the server that hosts the batch dispatcher.

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection factory}
queueRef={reference to a configured JMS queue} />
```

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher />`. You still must configure the **batchConnectionFactory** JMS connection factory and the JMS **batchJobSubmissionQueue** queue in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration. The following example illustrates the batch JMS dispatcher configuration and its JMS configuration.

**Note:** The **remoteServerAddress** attribute points to the `host:port` of the server that is hosting the Liberty message engine.

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
queueRef="batchJobSubmissionQueue" />

<jmsConnectionFactory id="batchConnectionFactory"
jndiName="jms/batch/connectionFactory">
 <properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id="batchJobSubmissionQueue"
jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
queueName="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>
```

2. Configure the batch JMS executor.
  - a. Enable JMS support by adding the **wasJmsClient-2.0** feature to the feature manager in your `server.xml`.
  - b. Add the **batchJmsExecutor** element to your `server.xml` file on the server that hosts the batch executor.

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
queueRef={reference to the configured JMS queue} />
```

**Note:** If you do not specify the `activationSpecRef` and `queueRef` attributes, the default value for `activationSpecRef` is **batchActivationSpec** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsExecutor` element as `<batchJmsExecutor />`. You still must configure the JMS activation specification for `batchActivationSpec` and the `batchJobSubmissionQueue` JMS queue in the `server.xml` file.

- c. Add the corresponding JMS activation specification and JMS queue to the server configuration. This is not specific to batch configuration.
  - d. Define batch executor server capabilities by including a JMS message selector in the activation specification.
    - Filtering based on system defined properties:  
There is a set of batch dispatcher properties available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com\_ibm\_ws\_batch\_applicationName**: the name of the batch application for the job request
- **com\_ibm\_ws\_batch\_moduleName**: the module name of the batch application for the job request
- **com\_ibm\_ws\_batch\_componentName**: the component name of the batch application for the job request

**Note:** It is recommended that a message selector is specified with at least the **com\_ibm\_ws\_batch\_applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application SimpleBatchJob and BonusPayout.

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'>
```

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application SimpleBatchJob.

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob'>
```

- Filtering based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method **Character.isJavaLetter** returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method **Character.isJavaLetterOrDigit** returns true. Check JMS Javadoc for more information on JMS message selectory.

The following example illustrates a possible message selector using the **com\_ibm\_ws\_batch\_applicationName** property and a job parameter **specialCapability**.

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'>
```

The following example illustrates the batch JMS executor configuration and its JMS configuration.

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"/>

<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"
 messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayoutCDI'"
 destinationType="javax.jms.Queue"
 remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
 queueName="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>
```

3. Install your batch application on the server. For more information, see Chapter 9, “Deploying applications in Liberty,” on page 1329.

## Enabling multiple server support by using the WebSphere MQ messaging provider



8.5.5.6

The batch environment can be set up to have servers function as batch dispatchers while other servers function as batch executors. Batch dispatchers accept requests from external clients and make them available to the batch executors. The batch executors receive requests that match its defined capabilities and execute those requests. Batch dispatchers and batch executors communicate by using Java Messaging Service (JMS).

## About this task

This task helps you configure the batch dispatch server and the batch executor by using the WebSphere MQ messaging provider.

### Procedure

1. Configure the batch JMS dispatcher.
  - a. Enable JMS support by adding the `wmqJmsClient-2.0` feature to the feature manager in your `server.xml` file.
  - b. Add the `batchJmsDispatcher` element to your `server.xml` file on the server that hosts the batch dispatcher.

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection factory}
queueRef={reference to a configured JMS queue} />
```

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is `batchConnectionFactory` and the default value for `queueRef` is `batchJobSubmissionQueue`. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher />`. You still must configure the `batchConnectionFactory` JMS connection factory and the JMS `batchJobSubmissionQueue` queue in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration.

The following example illustrates the batch JMS dispatcher configuration and its JMS configuration using WMQ binding mode.

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
queueRef="batchJobSubmissionQueue" />
```

```
<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
value="{server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
startupRetryCount=999
startupRetryInterval="1000ms"
reconnectionRetryCount=10
reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
jndiName="jms/batch/connectionFactory">
<properties.wmqJms transportType="BINDINGS"
queueManager="WMQX">
</properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
jndiName="jms/batch/jobSubmissionQueue">
<properties.wmqJms baseQueueName="BATCHQ"
priority="QDEF"
baseQueueManagerName="WMQX">
</properties.wmqJms>
</jmsQueue>
```

The following example illustrates the batch JMS dispatcher configuration and its JMS configuration using WMQ in client mode.

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
value="{server.config.dir}/wmq.wlp.rar"/>
```

```

<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory">
 <properties.wmqJms
 hostname="webs24.pok.stglabs.ibm.com"
 transportType="CLIENT"
 channel="WAS.JMS.SVRCONN"
 port="1414"
 queueManager="WMQX"/>>
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

## 2. Configure the batch JMS executor.

- a. Enable JMS support by adding the **wmqJmsClient-2.0** feature to the feature manager in your `server.xml` file.
- b. Add the **batchJmsExecutor** element to your `server.xml` file on the server that hosts the batch executor.

```

<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
 queueRef={reference to the configured JMS queue} />

```

**Note:** If you do not specify the `activationSpecRef` and `queueRef` attributes, the default value for `activationSpecRef` is **batchActivationSpec** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsExecutor` element as `<batchJmsExecutor />`. You still must configure the JMS activation specification for `batchActivationSpec` and the `batchJobSubmissionQueue` JMS queue in the `server.xml` file.

- c. Add the corresponding JMS activation specification and JMS queue to the server configuration. This is not specific to batch configuration.
- d. Define batch executor server capabilities by including a JMS message selector in the activation specification.

- Filtering based on system defined properties:

There is a set of batch dispatcher properties available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com\_ibm\_ws\_batch\_applicationName:** the name of the batch application for the job request
- **com\_ibm\_ws\_batch\_moduleName:** the module name of the batch application for the job request
- **com\_ibm\_ws\_batch\_componentName:** the component name of the batch application for the job request

**Note:** It is recommended that a message selector is specified with at least the **com\_ibm\_ws\_batch\_applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application `SimpleBatchJob` and `BonusPayout`.

```

messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'">

```

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application SimpleBatchJob.

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob'">
```

- Filtering based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method **Character.isJavaLetter** returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method **Character.isJavaLetterOrDigit** returns true. Check JMS Javadoc for more information on JMS message selectory.

The following example illustrates a possible message selector by using the **com\_ibm\_ws\_batch\_applicationName** property and a job parameter **specialCapability**.

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'">
```

The following example illustrates the batch JMS executor configuration and its JMS configuration using WMQ binding mode.

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
 startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 destinationType="javax.jms.Queue"
 messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'"
 transportType="BINDINGS"
 queueManager="WMQX">
 </properties.wmqJms>
</JmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>
```

The following example illustrates the batch JMS executor configuration and its JMS configurations using WMQ client mode.

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
 startupRetryInterval="1000ms"
```



```

 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 messageSelector="com_ibm_ws_batch_applicationName = SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'"
 transportType="CLIENT"
 channel="WAS.JMS.SVRCONN"
 destinationType="javax.jms.Queue"
 queueManager="WMQX"
 hostName="webs24.pok.stglabs.ibm.com"
 port="1414">
 </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

3. Install your batch application on the server. For more information, see Chapter 9, “Deploying applications in Liberty,” on page 1329.

## Enabling multiple server partitions support by using the WebSphere MQ messaging provider

8.5.5.8

You can set up the batch environment to have servers function as batch dispatchers, while other servers function as batch executors.

### About this task

Batch dispatchers accept requests from external clients and make them available to the batch executors. The batch executors receive requests that match its defined capabilities and execute those requests. If the job is configured to run partitions, the batch executors make them available to the other executors to run. The batch executors communicate by using Java Messaging Service (JMS). This task helps you configure the batch dispatch server and the batch executor by using the WebSphere MQ messaging provider.

### Procedure

1. Configure the batch JMS dispatcher.
  - a. Enable JMS support by adding the **wmqJmsClient-2.0** feature to the feature manager in your `server.xml` file.
  - b. Add the **batchJmsDispatcher** element to your `server.xml` file on the server that hosts the batch dispatcher; for example:

```

<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection factory}
 queueRef={reference to a configured JMS queue} />

```

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher/>`. You still must configure the **batchConnectionFactory** JMS connection factory and the JMS **batchJobSubmissionQueue** queue in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration.

The following example illustrates the batch JMS dispatcher configuration and its JMS configuration using WebSphere MQ binding mode:

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
 queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
 startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory">
 <properties.wmqJms transportType="BINDINGS"
 queueManager="WMQX">
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>
```

The following example illustrates the batch JMS dispatcher configuration and its JMS configuration using WebSphere MQ in client mode:

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
 queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory">
 <properties.wmqJms
 hostname="webs24.pok.stglabs.ibm.com"
 transportType="CLIENT"
 channel="WAS.JMS.SVRCONN"
 port="1414"
 queueManager="WMQX"/>
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>
```

## 2. Configure the batch JMS executor for executing jobs

- a. Enable JMS support by adding the **wmqJmsClient-2.0** feature to the feature manager in your `server.xml` file.
- b. Add the **batchJmsDispatcher** element to your `server.xml` file on the server that hosts the batch executor.

**Note:** If you do not add the **batchJmsDispatcher** element to your `server.xml` file, the server will not dispatch partitions to run on multiple servers and will run the partitions locally.

- c. Add the **batchJmsExecutor** element and a `connectionFactory` to your `server.xml` file on the server that hosts the batch executor. For more information, see step 1b and 1c.

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
queueRef={reference to the configured JMS queue} />
```

**Note:** If you do not specify the `activationSpecRef` and `queueRef` attributes, the default value for `activationSpecRef` is **batchActivationSpec** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsExecutor` element as `<batchJmsExecutor/>`. You still must configure the JMS activation specification for `batchActivationSpec` and the `batchJobSubmissionQueue` JMS queue in the `server.xml` file.

- d. Add the corresponding JMS activation specification and JMS queue to the server configuration. This is not specific to batch configuration.
- e. Define batch executor server capabilities by including a JMS message selector in the activation specification.

- Filter based on system-defined properties:

The following batch dispatcher properties are available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com\_ibm\_ws\_batch\_applicationName:** the name of the batch application for the job request
- **com\_ibm\_ws\_batch\_moduleName:** the module name of the batch application for the job request
- **com\_ibm\_ws\_batch\_componentName:** the component name of the batch application for the job request
- **com\_ibm\_ws\_batch\_work\_type:** the work type : "Job"

**Note:** Specify a message selector with at least the **com\_ibm\_ws\_batch\_applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application `SimpleBatchJob` and `BonusPayout`:

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'>
```

- Filter based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method **Character.isJavaLetter** returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method **Character.isJavaLetterOrDigit** returns true. Check the JMS API documentation for more information on JMS message selectory.

The following example illustrates a possible message selector by using the **com\_ibm\_ws\_batch\_applicationName** property and a job parameter **specialCapability**:

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'>
```

The following example illustrates the batch JMS executor configuration and its JMS configuration using WebSphere MQ binding mode:

```

<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
 startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 destinationType="javax.jms.Queue"
 messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_ws_batch_applicationName = 'BINDINGS'"
 transportType="BINDINGS"
 queueManager="WMQX">
 </properties.wmqJms>
</JmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

The following example illustrates the batch JMS executor configuration and its JMS configurations using Websphere MQ client mode:

```

<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 messageSelector="com_ibm_ws_batch_applicationName = SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout'"
 transportType="CLIENT"
 channel="WAS.JMS.SVRCONN"
 destinationType="javax.jms.Queue"
 queueManager="WMQX"
 hostName="webs24.pok.stglabs.ibm.com"
 port="1414">
 </properties.wmqJms>
</JmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

3. Configure the batch JMS executor for executing only partitions.
  - a. Enable JMS support by adding the **wmqJmsClient-2.0** feature to the feature manager in your `server.xml` file.
  - b. Add the **batchJmsExecutor** element and a `connectionFactory` to your `server.xml` file on the server that hosts the batch executor. For more information, see step 1b and 1c.

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
 queueRef={reference to the configured JMS queue}
 replyConnectionFactoryRef={reference to the configured JMS connection factory} />
```

**Note:** If you do not specify the `activationSpecRef` and `queueRef` attributes, the default value for `activationSpecRef` is **batchActivationSpec** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsExecutor` element as `<batchJmsExecutor/>`. You still must configure the JMS activation specification for `batchActivationSpec` and the `batchJobSubmissionQueue` JMS queue in the `server.xml` file.

- c. Add the corresponding JMS activation specification and JMS queue to the server configuration. This is not specific to batch configuration.
- d. Define batch executor server capabilities by including a JMS message selector in the activation specification.

- Filter based on system-defined properties:

The following batch dispatcher properties are available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com.ibm.ws.batch.applicationName:** the name of the batch application for the job request
- **com.ibm.ws.batch.moduleName:** the module name of the batch application for the job request
- **com.ibm.ws.batch.componentName:** the component name of the batch application for the job request
- **com.ibm.ws.batch.work.type:** the work type : "Partition"
- **com.ibm.ws.batch.partitionNum**(Type=Integer): the partition number of the partition that is being dispatched
- **com.ibm.ws.batch.stepName:** the name of the step as defined in the JSL

**Note:** Specify a message selector with at least the **com.ibm.ws.batch.applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept a job for the application `SimpleBatchJob` and `BonusPayout`:

```
messageSelector="com.ibm.ws.batch.applicationName = 'SimpleBatchJob' OR com.ibm.ws.batch.applicationName = 'BonusPayout'>
```

- Filter based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method **Character.isJavaLetter** returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method **Character.isJavaLetterOrDigit** returns true. Check the JMS API documentation for more information on JMS message selector.

The following example illustrates a possible message selector by using the **com.ibm.ws.batch.applicationName** property and a job parameter **specialCapability**:

```
messageSelector="com.ibm.ws.batch.applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'>
```

The following example illustrates the batch JMS executor configuration and its JMS configuration using WebSphere MQ binding mode:

```
<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory"/>
```

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
```

```

 queueRef="batchJobSubmissionQueue"
 replyConnectionFactoryRef="batchConnectionFactory"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
 startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 destinationType="javax.jms.Queue"
 messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_ws_batch_applicationName = 'BonusPayout'"
 transportType="BINDINGS"
 queueManager="WMQX">
 </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wmqJms baseQueueName="BATCHQ"
 priority="QDEF"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

The following example illustrates the batch JMS executor configuration and its JMS configurations using Webs Sphere MQ client mode:

```

<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory"/>

<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"
 replyConnectionFactoryRef="batchConnectionFactory"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
 value="{server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
 recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
 startupRetryInterval="1000ms"
 reconnectionRetryCount=10
 reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
 <properties.wmqJms destinationRef="batchJobSubmissionQueue"
 messageSelector="(com_ibm_ws_batch_applicationName = SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_ws_batch_applicationName = 'BonusPayout'"
 transportType="CLIENT"
 channel="WAS.JMS.SVRCONN"
 destinationType="javax.jms.Queue"
 queueManager="WMQX"
 hostName="webs24.pok.stglabs.ibm.com"
 port="1414">
 </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">

```

```

 <properties.wmqJms baseQueueName="BATCHQ"
 baseQueueManagerName="WMQX">
 </properties.wmqJms>
</jmsQueue>

```

4. Install your batch application on the server. For more information, see Chapter 9, “Deploying applications in Liberty,” on page 1329. To disable multiple server partition execution: If you want to have multiple server support but do not want multiple server partition executions, you can set job property `com.ibm.websphere.batch.partition.multiJVM` to `false` in your `jsl` job XML file. The following example illustrates a JSL job to disable multiple server partitions:

```

<property name="com.ibm.websphere.batch.partition.multiJVM" value="false"/>

```

**Note:** If you are running a partition on a remote executor, there are no job logs created for that partition.

## Enabling multiple server partitions support by using the Liberty embedded messaging provider

8.5.5.8

You can set up the batch environment to have servers function as batch dispatchers, while other servers function as batch executors.

### Before you begin

1. Determine where the embedded message engine is hosted. It can be hosted on the batch dispatch server, the batch executor server, or on a separate server. This server must be configured before you complete this task. The JMS connection factory and activation specification references the message engine server in its configuration.
2. To configure the message engine:
  - a. Add the **wasJmsServer-1.0** feature to the `server.xml`.
  - b. Define the message engine by adding the **messageEngine** element. Define the queue that is used for the batch dispatcher and batch executor. The following example illustrates the message engine configuration in your `server.xml` file:

```

<!--specify the ports for the message engine.
The ports in this example are the default ports.
This element is not needed when the default ports are used. -->
<wasJmsEndpoint host="*"
 wasJmsPort="7280"
 wasJmsSSLPort="7290"
 enabled="true">
</wasJmsEndpoint>

<messagingEngine>
 <!-- queue for batch jms message. -->
 <queue id="batchLibertyQueue"
 forceReliability="ReliablePersistent"
 receiveAllowed="true"/>
</messagingEngine>

```

### About this task

Batch dispatchers accept requests from external clients and make them available to the batch executors. The batch executors receive requests that match its defined capabilities and execute those requests. If the job is configured to run partitions, the batch executors make them available to the other executors to run. The batch executors communicate with each other by using Java Messaging Service (JMS). This task helps you configure the batch dispatch server and the batch executor by using the Liberty profile embedded messaging provider.

### Procedure

1. Configure the batch JMS dispatcher.

- a. Enable JMS support by adding the **wasJmsClient-2.0** feature to the feature manager in your `server.xml` file.
- b. Add the **batchJmsDispatcher** element to your `server.xml` file on the server that hosts the batch dispatcher; for example:

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection factory}
queueRef={reference to a configured JMS queue} />
```

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher/>`. You still must configure the **batchConnectionFactory** JMS connection factory and the JMS **batchJobSubmissionQueue** queue in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration. The following example illustrates the batch JMS dispatcher configuration and its JMS configuration:

**Note:** The `remoteServerAddress` attribute points to the `host:port` of the server that is hosting the Liberty profile message engine.

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
queueRef="batchJobSubmissionQueue" />

<jmsConnectionFactory id="batchConnectionFactory"
jndiName="jms/batch/connectionFactory">
 <properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id="batchJobSubmissionQueue"
jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
queuename="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>
```

## 2. Configure the batch JMS executor for executing batch jobs.

- a. Enable JMS support by adding the **wasJmsClient-2.0** feature to the feature manager in your `server.xml` file.
- b. Add the **batchJmsDispatcher** element to your `server.xml` file on the server that hosts the batch executor.

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
queueRef={reference to the configured JMS queue} />
```

**Note:** If you do not add the **batchJmsDispatcher** element to your `server.xml` file, the server will not dispatch partitions to run on multiple servers and will run the partitions locally.

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher/>`. You still must configure the **batchConnectionFactory** JMS connections factory and the `batchJobSubmissionQueue` in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration. The following example illustrates the batch JMS dispatcher configuration and its JMS configuration:

**Note:** The `remoteServerAddress` attribute points to the `host:port` of the server that is hosting the Liberty profile message engine.

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
queueRef="batchJobSubmissionQueue" />

<jmsConnectionFactory id="batchConnectionFactory"
jndiName="jms/batch/connectionFactory">
```



```

<properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
</properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id='batchJobSubmissionQueue'
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
 queueName="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>

```

- d. Add the **batchJmsExecutor** element to your **server.xml** on the server that hosts the batch executor; for example:

```

<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
 queueRef={reference to the configured JMS queue} />

```

**Note:** If you do not specify the `activationSpecRef` and `queueRef` attributes, the default value for `activationSpecRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsExecutor` element as `<batchJmsExecutor/>`. You still must configure the JMS activation specification for **batchActivationSpec** and the **batchJobSubmissionQueue** queue in the `server.xml` file.

- e. Add the corresponding JMS activation specification and JMS queue to the server configuration. This is not specific to batch configuration.
- f. Define batch executor server capabilities by including a JMS message selector in the activation specification.

- Filter based on system-defined properties:

The following batch dispatcher properties are available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com\_ibm\_ws\_batch\_applicationName**: the name of the batch application for the job request
- **com\_ibm\_ws\_batch\_moduleName**: the module name of the batch application for the job request
- **com\_ibm\_ws\_batch\_componentName**: the component name of the batch application for the job request
- **com\_ibm\_ws\_batch\_work\_type**: the work type : "Job"

**Note:** Specify a message selector with at least the **com\_ibm\_ws\_batch\_applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept only batch jobs for the application `SimpleBatchJob`:

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND com_ibm_ws_batch_work_type = 'Job'">
```

- Filtering based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method `Character.isJavaLetter` returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method `Character.isJavaLetterOrDigit` returns true. Check the JMS API documentation for more information on JMS message selector.

The following example illustrates a possible message selector by using the **com\_ibm\_ws\_batch\_applicationName** property and a job parameter **specialCapability**:

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'">
```

The following example illustrates the batch JMS executor configuration and its JMS configuration:

```

<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"/>

<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"

```

```

 messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR com_ibm_ws_batch_applicationName = 'BonusPayoutCDI') AND com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND com_ibm_ws_batch_destinationType='javax.jms.Queue' AND com_ibm_ws_batch_remoteServerAddress='host:7280:BootstrapBasicMessaging'"
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
 queueName="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>

```

### 3. Configure the batch JMS executor for executing only partitions.

- a. Enable JMS support by adding the **wasJmsClient-2.0** feature to the feature manager in your `server.xml` file.
- b. Add the **batchJmsExecutor** element to your `server.xml` file on the server that hosts the batch executor that is running batch jobs.

```

<batchJmsExecutor activationSpecRef={configured activation specification or batch executor}
 queueRef={reference to the configured JMS queue}
 replyConnectionFactoryRef={reference to the configured JMS connection factory} />

```

**Note:** If you do not specify the `connectionFactoryRef` and `queueRef` attributes, the default value for `connectionFactoryRef` is **batchConnectionFactory** and the default value for `queueRef` is **batchJobSubmissionQueue**. You can specify the `batchJmsDispatcher` element as `<batchJmsDispatcher/>`. You still must configure the **batchConnectionFactory** JMS connections factory and the `batchJobSubmissionQueue` in the `server.xml` file.

- c. Add the corresponding JMS connection factory and JMS queue to the server configuration. This is not specific to batch configuration.
- d. Define batch executor server capabilities by including a JMS message selector in the activation specification.
  - Filter based on system-defined properties:

The following batch dispatcher properties are available on the batch JMS message that the batch executor can use to filter for inbound messages.

- **com\_ibm\_ws\_batch\_applicationName:** the name of the batch application for the job request
- **com\_ibm\_ws\_batch\_moduleName:** the module name of the batch application for the job request
- **com\_ibm\_ws\_batch\_componentName:** the component name of the batch application for the job request
- **com\_ibm\_ws\_batch\_work\_type:** the work type : "Partition"
- **com\_ibm\_ws\_batch\_partitionNum**(Type=Integer): the partition number of the partition that is being dispatched
- **com\_ibm\_ws\_batch\_stepName:** the name of the step as defined in the JSL

**Note:** Specify a message selector with at least the **com\_ibm\_ws\_batch\_applicationName** property to ensure that the executor only receives jobs that it can process.

The following example indicates the **messageSelector** attribute for the executor to accept only partitions for the application `SimpleBatchJob`:

```

messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND com_ibm_ws_batch_work_type = 'Partition'"

```

The following example indicates the **messageSelector** attribute for the executor to accept only partitions for the application `SimpleBatchJob`:

```

messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND com_ibm_ws_batch_work_type = 'Partition' AND com_ibm_ws_batch_stepName = 'SimpleBatchJob'"

```

- Filtering based on user-defined properties:

The batch dispatcher sets all job parameters that conform to the proper JMS message property on the batch dispatcher request message. These properties can also be used by the message selector to add extra filtering to the message selector. The property name, or identifier, must conform to JMS message property constraints. For example, the property is an unlimited length

sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method `Character.isJavaLetter` returns true, and includes '\_' and '\$'. A letter or digit is any character for which the method `Character.isJavaLetterOrDigit` returns true. Check the JMS API documentation for more information on JMS message selectory.

The following example illustrates a possible message selector by using the `com.ibm.ws.batch.applicationName` property and a job parameter `specialCapability`:

```
messageSelector="com.ibm.ws.batch.applicationName = 'SimpleBatchJob' AND specialCapability = 'superCapability'">
```

The following example illustrates the batch JMS executor configuration and its JMS configuration:

```
<jmsConnectionFactory id="batchConnectionFactory"
 jndiName="jms/batch/connectionFactory"/>

<batchJmsExecutor activationSpecRef="batchActivationSpec"
 queueRef="batchJobSubmissionQueue"
 replyConnectionFactoryRef="batchConnectionFactory"/>

<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"
 messageSelector="(com.ibm.ws.batch.applicationName = 'SimpleBatchJob' OR com.ibm.ws.batch.applicationName = 'BonusPayoutCDI') AND com_i
 destinationRef="batchJobSubmissionQueue"
 destinationType="javax.jms.Queue"
 remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
 jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
 queueName="batchLibertyQueue">
 </properties.wasJms>
</jmsQueue>
```

4. Install your batch application on the server. For more information, see Chapter 9, “Deploying applications in Liberty,” on page 1329. To disable multiple server partition execution: If you want to have multiple server support but do not want multiple server partition executions, you can set job property `com.ibm.websphere.batch.partition.multiJVM` to false in your js1 job XML file. The following example illustrates a JSL job to disable multiple server partitions:

```
<property name="com.ibm.websphere.batch.partition.multiJVM" value="false"/>
```

**Note:** If you are running a partition on a remote executor, there are no job logs created for that partition.

## Enabling batch job events publishing

8.5.5.7

By using Java Messaging System (JMS), the batch server can publish job-related events to external clients.

### About this task

The ability of the batch server to publish job-related events to external clients enables a monitor to see job-related events and report on failures. The batch dispatcher server can publish events for a job in the dispatching phase. The batch executor server can publish events for jobs when it moves through different phases of execution. These events are published in a topic tree in the following structures:

Table 96. File structures of topic trees for event publishing

Structure	Description
batch	The root of the topic tree.
batch/jobs	The topic tree for all job-related events.
batch/jobs/instance	The topic tree for all events that are related to a job instance.

Table 96. File structures of topic trees for event publishing (continued)

Structure	Description
batch/jobs/instance/submitted	A topic tree node. A message is published when the batch server creates a job instance for a new job submission.
batch/jobs/instance/jms_queued	A topic tree node. A message is published when job submission is placed on the job submission queue by the batch JMS dispatcher.
batch/jobs/instance/jms_consumed	A topic tree node. A message is published when the batch executor receives the job submission from the job submission queue.
batch/jobs/instance/dispatched	A topic tree node. A message is published when the batch executor accepts a job instance for execution.
batch/jobs/instance/completed	A topic tree node. A message is published when the job instance is completed.
batch/jobs/instance/stopped	A topic tree node. A message is published when the job instance is stopped.
batch/jobs/instance/stopping	A topic tree node. A message is published when the job instance is stopping.
batch/jobs/instance/failed	A topic tree node. A message is published when the job instance failed.
batch/jobs/instance/purged	A topic tree node. A message is published when a job instance is purged successfully.
batch/jobs/execution	The topic tree for all events that are related to a job execution.
batch/jobs/execution/restarting	A topic tree node. A message is published when the batch executor is restarting an execution.
batch/jobs/execution/starting	A topic tree node. A message is published when a job execution is starting.
batch/jobs/execution/completed	A topic tree node. A message is published when a job execution ends successfully.
batch/jobs/execution/failed	A topic tree node. A message is published when a job execution ends because of failure.
batch/jobs/execution/stopped	A topic tree node. A message is published when a job execution is stopped.
batch/jobs/execution/jobLogPart	A topic tree node. A message is published when a new job log part is created, a job stops, or a job ends.
batch/jobs/execution/step/started	A topic tree node. A message is published when a step execution is started.
batch/jobs/execution/step/completed	A topic tree node. A message is published when a step execution is completed successfully.
batch/jobs/execution/step/failed	A topic tree node. A message is published when a step execution fails.
batch/jobs/execution/step/stopped	A topic tree node. A message is published when a step execution is stopped.

Table 96. File structures of topic trees for event publishing (continued)

Structure	Description
batch/jobs/execution/step/checkpoint	A topic tree node. A message is published when a checkpoint is taken.
batch/jobs/execution/partition/started	A topic tree node. A message is published when a partition is started.
batch/jobs/execution/partition/completed	A topic tree node. A message is published when a partition is completed successfully.
batch/jobs/execution/partition/failed	A topic tree node. A message is published when a partition fails.
batch/jobs/execution/partition/stopped	A topic tree node. A message is published when a partition is stopped.
batch/jobs/execution/split-flow/started	A topic tree node. A message is published when a split-flow is started.
batch/jobs/execution/split-flow/ended	A topic tree node. A message is published when a split-flow is completed.

The published message for each topic is a JMS TextMessage. The contents of this message is a JSON formatted string that represents the object of the topic, such as job instance, job execution, step execution, or partition. In addition, this message also includes the following JMS message properties set:

- com\_ibm\_ws\_batch\_internal\_jobInstanceId: The job instance ID, if available.
- com\_ibm\_ws\_batch\_internal\_jobExecutionId: The job execution ID, if available.
- com\_ibm\_ws\_batch\_internal\_stepExecutionId: The job step execution ID, if available.

The batch server must be configured to enable the publishing of job-related events. The batch dispatcher and batch executor have the same configuration. The following steps enable the publication of job-related events for a batch server.

## Procedure

1. Enable JMS support by adding the appropriate JMS feature to the feature manager in the server.xml file. If you are using the WebSphere Application Server Liberty default messaging provider, add the **wasJmsClient-2.0** feature and related JMS configurations for the message engine. If you are using WebSphere MQ Messaging provider, add the **wmqJmsClient-2.0** feature.

2. Add the **batchJmsEvents** element to the server.xml file.

```
<batchJmsEvents connectionFactoryRef="batchConnectionFactory" />
```

**Note:** If you do not specify the **connectionFactoryRef** attribute, the default value for **connectionFactoryRef** is **batchConnectionFactory**. You must still configure the **batchConnectionFactory** JMS connection factory in the server.xml file.

3. Add the corresponding JMS connection factory to the server configuration. This is not specific to batch configuration.

The following example illustrates the batch events configuration and its JMS configuration by using the WebSphere MQ messaging provider.

```
<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location" value="{server.config.dir}/wmq.wlp.rar"/>
<!-- require for BINDING mode -->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"/>

<batchJmsEvents connectionFactoryRef="batchConnectionFactory" />
```

```

<jmsConnectionFactory id="batchConnectionFactory" jndiName="jms/batch/connectionFactory">
 <properties.wmq.Jms
 transportType="BINDINGS"
 queueManager="WMQX" />
</jmsConnectionFactory>

```

The following example illustrates the batch events configuration and its JMS configuration by using the WebSphere Liberty default messaging provider.

```

<batchJmsEvents connectionFactoryRef="batchConnectionFactory" />
<jmsConnectionFactory id="batchConnectionFactory" jndiName="jms/batch/connectionFactory">
 <properties.wasJms></properties.wasJms>
</jmsConnectionFactory>

```

## Example

The following examples illustrate sequence of events for basic execution flows.

- Submit and run a single-step job with checkpoints.

```

batch/jobs/instance/submitted
batch/jobs/instance/jms_queued
batch/jobs/instance/jms_consumed
batch/jobs/execution/starting
batch/jobs/instance/dispatched
batch/jobs/execution/started
batch/jobs/execution/step/started
batch/jobs/execution/step/checkpoint
batch/jobs/execution/step/checkpoint
...
batch/jobs/execution/step/checkpoint
batch/jobs/execution/step/completed
batch/jobs/execution/completed
batch/jobs/instance/completed

```

- Submit and run a single-step job with partition.

```

batch/jobs/instance/submitted
batch/jobs/instance/jms_queued
batch/jobs/instance/jms_consumed
batch/jobs/execution/starting
batch/jobs/instance/dispatched
batch/jobs/execution/started
batch/jobs/execution/step/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/completed
batch/jobs/execution/partition/completed
batch/jobs/execution/partition/completed
batch/jobs/execution/step/completed
batch/jobs/execution/completed
batch/jobs/instance/completed

```

## batchManager command-line client utility



8.5.5.6

The batchManager command-line client utility provides a command-line interface for managing your batch jobs that run on Liberty.

The batchManager command-line client utility interacts with the batch manager over the batch manager's REST API. To use the batchManager command-line client utility, the batch manager must be running on your Liberty server. Use the batch management feature to install and enable the Liberty batch manager.

## SSL configuration

The batchManager command-line client utility communicates with the batch manager over an SSL connection. To facilitate SSL communication with a batch manager that is running on a Liberty server, the utility must be able to verify the SSL certificate of the Liberty server.

If the SSL certificate is signed by a well-known certificate authority (CA), the utility can verify the certificate by the CA. No further configuration is necessary.

If the SSL certificate is not signed by a CA, then you must configure the utility to trust the SSL certificate of the server by doing one of the following actions.

- Specify the option `--trustSslCertificates`, which configures the utility to trust all SSL certificates.
- Include the server's SSL certificate in the utility's truststore.

If you choose to specify the option `--trustSslCertificates`, the utility trusts all SSL certificates that it receives and no further configuration is necessary.

If you choose the option to include the server's SSL certificate in the utility's truststore, then you must also configure the utility so that it can find its truststore. The utility is a stand-alone Java main. You configure SSL by using system properties such as `javax.net.ssl.truststore`.

If the batch manager is running on the same machine as the utility, then you can point the utility directly at the server keystore:

```
$ export JVM_ARGS="-Djavax.net.ssl.trustStore=/path/to/server/keystore.jks"
$ batchManager submit ...
```

**Attention:** JVM arguments, such as `-D` properties, are passed to the batchManager command-line client utility by the `JVM-ARGS` environment variable.

If you cannot use the server keystore directly, you must export the server certificate from the server keystore and import it into the client truststore. Use the JDK keytool utility for exporting and importing certificates. In the following example, the server certificate is stored in the `[server-dir]/resources/security/key.jks` keystore file under the `default` alias, and the password is `Liberty`.

```
$ keytool -export -alias default -file server.crt -keystore [server-dir]/resources/security/key.jks -storepass Liberty
$ keytool -import -alias server_cert -file server.crt -keystore /path/to/truststore.jks -storepass password
```

**Attention:** The `import` command creates the `truststore.jks` file if the file does not exist.

```
$ export JVM_ARGS="-Djavax.net.ssl.trustStore=/path/to/truststore.jks"
$ batchManager submit ...
```

## Commands and usage

The batchManager command-line client utility provides commands for submitting, stopping, restarting, and checking the status of jobs.

To generally use the utility:

```
$ batchManager [command] [options]
```

To see a list of available commands:

```
$ batchManager help
```

To see the description and options for a specific command:

```
$ batchManager help [command]
```

The following example illustrates how to submit a job and wait for its completion:

```
$ batchManager submit \
 --batchManager=<host>:<port>
 --user=[credentials for logging into the batch manager]
 --password=[credentials for logging into the batch manager]
 --applicationName=[application name used when packaging the batch app]
 --jobXMLName=[job XML file basename in the app's batch-jobs dir]
 --wait
```

## jobParametersFile and jobPropertiesFile

When submitting a batch job by using the batchManagerZos client utility, the jobParametersFile and jobPropertiesFiles supports the use of multiple files separated by commas. Files later in the comma separated list take precedence over files that appear first in the list. The following example illustrates correct usage of the comma separated list.

```
jobParametersFile=filePath1,filePath2,filePath3
jobPropertiesFile=filePath1,filePath2,filePath3
```

As an example, `--jobParametersFile=<filepath1>` would override `--jobParametersFile=<filepath1>,<filepath2>` in the control properties file. The resulting parameter is `--jobParametersFile=<filepath1>`.

## Return codes

The batchManager command-line client utility outputs the following return codes:

Code	Description
0	The task completed normally.
20	A required argument was not specified.
21	An unrecognized argument was specified.
22	An invalid argument value was specified.
255	An unknown error occurred.

**Note:** If you specify the `--wait` argument, the utility outputs the following return codes about the status of the job that you are waiting for.

Code	Description
33	The job stopped.
34	The job did not complete successfully.
35	The job completed successfully.
36	The job was abandoned.

## Viewing Java batch job logs



8.5.5.6

When you are running Java batch jobs in the WebSphere Application Server Liberty, a log is written for each job.

The logs are created in the following directory structure:



*log directory/joblogs/job name/date/instance.job instance ID/execution.execution ID*

**Attention:** The variable *job name* is the id attribute of the job element within the JSL (XML) document. It is not necessarily related to the file name of the JSL file.

The naming of logs begins at part.1.log and rotates to new log parts as needed. By default, the job log contains all messages and trace information that is logged in the server by the thread that performs the job execution. Output that is not logged within the `java.util.logging` framework is not collected.

**Important:** If a new log part cannot be created, batch attempts to continue processing without a log, or with the current log part.

For more information about retrieving or deleting job logs by using the REST API, see the *REST API administration* documentation.

**Important:** If you use the log4j API, applications that use the log4j framework can participate in batch job logging by using `org.apache.log4j.jul.JULAppender`. The `JULAppender` forwards log4j log records to the `java.util.logging` framework, where they are collected for job logging.

## Configuring job logging

Batch job logging can be configured by using the `<batchJobLogging>` configuration element `<batchJobLogging enabled="true" maxRecords="1000" />`.

The attribute `maxRecords` indicates the number of records that are written to a job log part before the records roll over to the next part.

The batch feature uses a logger that is named `com.ibm.ws.batch.JobLogger` to log certain batch messages to the job log only. Examples include job lifecycle messages and checkpoint messages. The logger does not write to the server log. By default, the logger is enabled for Level `.FINE` messages. You can configure the level of the logger by specifying it in the trace specification of the server. For example, `<logging traceSpecification="*=info:com.ibm.ws.batch.JobLogger=all" />`.

Any log messages that are written by the job thread, including messages that are written by the run time and by application code, are written to both the job log and the server log.

The `System.out` and `System.err` files are only written to the server log and are not written to the job log.

## Partitioned steps

Partitioned steps have more subdirectories for each partition. The log files in the *execution ID* directory contain entries from the thread that is running the top-level job. The job logs for the partitions are stored with the following structure:

*log directory/joblogs/job name/date/instance.job instance ID/execution.execution ID/name of partitioned step/partition number*

**Important:** A directory is created in the *name of partitioned step* directory for each partition.

## Split flows

If a split flow occurs in the job, more subdirectories are created to capture the output from the thread of each flow. The log files that are located directly under the *execution ID* directory contains entries from the thread that is running the top-level job. The job logs for individual flow threads are stored with the following structure:

*log directory/joblogs/job name/date/instance.job instance ID/execution.execution ID/split ID/flow ID*

**Important:** A directory is created in the *split ID* directory for each flow.

## Job log events

If batch job events are enabled, then job log events are published when a job log part is completed and when a job goes to an ended state, such as stopped, failed, or completed. The job log event messages contain multiple JSON properties to help with identifying the message along with the actual job log file content.

**Important:** The JSON properties are part of the JMS message body and can be retrieved by converting the body text to a `JsonObject` and pulling the specific JSON property from that object.

The following example illustrates how to retrieve the job log content property.

```
//The retrieved job log event message
Message msg
//Convert the Message to a TextMessage
TextMessage txtMsg = (TextMessage) msg;
//Convert the text in the message to a JsonObject
JsonObject jobLogEventObject = Json.createReader(new StringReader(txtMsg.getText())).readObject();
//Pull the job log text content from the JsonObject as an example
JSONArray logContentArray = jobLogEventObject.getJSONArray("contents");
```

For more information about enabling batch job events, see *Enabling batch job events publishing*.

---

## Shared libraries

Shared libraries are files used by multiple applications. You can use shared libraries and global libraries to reduce the number of duplicate library files on your system.

### Library elements

Liberty libraries have three elements; `<folder>`, `<file>`, and `<fileset>`. For example:

```
<library>
 <folder dir="..." />
 <file name="..." />
 <fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>
```

A specified file must be a container for the resource (for example a JAR file) rather than the resource itself.

If an element in the list is a file, the contents of that JAR or compressed .zip file are searched. If a folder is specified then resources are loaded from that directory.

### Global libraries

Global libraries can be used by any application. JAR files are placed in a global library directory, and then are specified in the class loader configuration for each application.

You can place global libraries in two locations:

- `${shared.config.dir}/lib/global`
- `${server.config.dir}/lib/global`

If there are files present in these locations at the time an application is started, and that application does not have a `<classloader>` element configured, the application uses these libraries. If a class loader configuration is present, these libraries are not used unless the global library is explicitly referenced.

For more information, see “Providing global libraries for all Java EE applications” on page 978.

## Resource files

Within Liberty libraries, you can have resource files defined in the library element. For example,

```
<library>
 <folder dir="..." />
 <file name="..." />
 <fileset dir="..." includes="*.jar" scanInterval="5s" />
 <folder dir="${server.config.dir}/mylibs" />
 <file name="${server.config.dir}/otherlibs/my.jar" />
</library>
```

The folder setting in the previous example, allows all files under the `mylibs` directory to be available on the classpath. You can use this style of entry to have your `.xml` and `.properties` available.

## Library elements

Liberty libraries have three child elements, `<folder>`, `<file>` and `<fileset>`. For example,

```
<library>
 <folder dir="..." />
 <file name="..." />
 <fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>
```

- `<folder>`: All resources under each configured folder will be loadable
- `<file>`: Each configured file should be either a native library or a container for resources (such as a JAR or a ZIP file). All resources within a container are loadable and any other filetype that is specified will have no effect.
- `<fileset>`: Each configured fileset is effectively a collection of files. Each file in the fileset should be a native library or a container for resources (such as a JAR or a ZIP file). All resources within a container are loadable and any other filetype that is specified will have no effect.

For example,

```
<library id="someLibrary">
 <!-- Location of XML and .properties files in the file system for easy editing -->
 <folder dir="${server.config.dir}/editableConfig" />

 <!-- Location of some classes and resources in the file system -->
 <folder dir="${server.config.dir}/extraStuff" />

 <!-- A zip file containing some resources -->
 <file name="${server.config.dir}/lib/someResources.zip" />

 <!-- All the jar files in ther servers lib folder -->
 <fileset dir="${server.config.dir}/lib" includes="*.jar" scanInterval="5s" />
</library>

<application location="webStore.war">
 <classloader commonLibraryRef="someLibrary" />
</application>
```

The configuration snippet in the previous example, allows all resources under the `editableConfig` directory to be loaded by the `webStore` application.

---

## Loose applications

8.5.5.4

Loose applications are applications that are composed from multiple physical locations, which are provided to the run time via an XML file. Loose applications are supported for Java EE and OSGi applications and are particularly beneficial in a development environment.

### Normal application

Normally an application is contained under one directory (or in one archive), with its content, modules, resources, classdata, and metadata at known locations within that directory. For example: the location of resources for a web application is as follows:

- Library Java archive (JAR) files are stored in `WEB-INF/lib`
- Classes are either in library JAR files or in `WEB-INF/classes`
- The deployment descriptor is in `WEB-INF/web.xml`
- Content to be served is located from the root of the directory

### Loose application

A loose application is described as “a virtual directory that represents the application, where information might be in any location”. It enables development tools, such as WebSphere Application Server Developer Tools, to run applications where the associated files are loaded directly from the workspace, rather than being exported. Examples of associated files are Java classes, JavaServer Pages, or images. If you load the associated files directly from the workspace, it results in a faster build-run-debugging cycle. The content is not present under one directory, but can come from other locations. These locations are specified in an XML configuration file.

There are two ways you can provide the XML file to the run time:

1. By placing the XML file in the `location` attribute in an application configuration element with an appended `.xml` suffix
2. By placing the XML file directly into the application `dropins` folder

For example, if you specify `<application location="myapp.war" />`, the run time looks for a file called `myapp.war.xml`. The search rules are the same as for an application directory or archive. If the application files and the `.xml` loose application configuration files are both found, then the loose application configuration file is ignored. For example, if you have `myapp.war` and `myapp.war.xml`, the Liberty server uses `myapp.war` to run the application. You can also deploy loose applications directly into the `dropins` folder. To use the `dropins` folder, follow the naming conventions that are defined for the folder and append `.xml` to the end of the file name.

### Loose application configuration file

The Liberty server uses the loose application configuration file to obtain the application content, rather than locating it from a root directory or single archive. Using the appropriate XML, you can take the following actions:

- Map any physical directory to any location within the application
- Map any physical file to any location within the application
- Map any physical JAR file or directory to any location as a nested archive
- Map multiple physical sources to a single target location (merging)

For example:

1. Map the root of the archive to one location on disk, such as a folder in an Eclipse project.

2. Map a Java bin/output folder that is not in the “usual” location into the WEB-INF/classes folder. This location might be in a different folder due to your workspace preferences, corporate guidelines, source control project layout guidelines, and so on. You might have multiple Java source and output locations in the same project, and want to map them both to WEB-INF/classes.
3. Map an “external” JAR file into the application. This “external” JAR file might be one of the following:
  - A separate Java project that you want to treat like a JAR file in WEB-INF/lib
  - A utility JAR file somewhere else on your hard disk drive that you built the .war file against, and need to include in WEB-INF/lib at run time

## Loose application configuration file examples

You can configure three different elements in the loose application configuration file:

- <archive> for archives
- <file> for files
- <dir> for directories

### Archives

The <archive> element is always used as the root of the loose application configuration file. It is also the root of the virtual file system that is represented in the XML. You can nest any of the three elements under the root <archive> element. The root <archive> element does not have any attributes.

Archive elements can be nested recursively. For <archive> elements nested under the root <archive> element, you can set the targetInArchive attribute. The targetInArchive attribute defines the path where the archive appears within the loose defined enclosing archive. You cannot map an archive on the file system as an archive in the application with an <archive> element. To use loose application configuration to map an archive on disk, use a <file> element instead.

**Note:** The targetInArchive attribute value is an absolute path with a leading forward slash (/).

The following code is an example of the root <archive> element with another <archive> element nested under it:

```
<archive>
 <archive targetInArchive="/jarName.jar">
 <!-- more objects can be embedded here-->
 </archive>
</archive>
```

**Files** You can use the <file> element to map a file on your hard disk to a file in your loose application configuration. You can set the following attributes on the <file> element:

- targetInArchive defines the path where the archive appears within the loose defined enclosing archive.
- sourceOnDisk defines the actual location of your file on your file system.

**Note:** The sourceOnDisk attribute value is an absolute location. You can use Liberty variables such as \${was.server.dir}, which are resolved correctly.

The following code is an example of a file in C:/devFolder/myApplication.zip that is represented as /apps/webApplication.war by the loose application configuration:

```
<file targetInArchive="/apps/webApplication.war"
 sourceOnDisk="C:/devFolder/myApplication.zip" />
```

The following code is an example of an enclosing archive that defines an archive at path /apps/webApplication.war. Within the archive, webApplication.war defines a file, jarName.jar at path /applications/myApplications. The actual file location is c:\devFolder\myApplication.zip:

```
<archive targetInArchive="/apps/webApplication.war">
 <file targetInArchive="/applications/myApplications/jarName.jar"
 sourceOnDisk="C:/devFolder/myApplication.zip" />
</archive>
```

## Directories

You can use the `<dir>` element to map a directory, and all of its contents on disk, to a directory location in the loose application configuration. The element has the same attributes as the `<file>` element and you use it in a similar way.

The following code is an example of a directory that the loose application configuration shows as being in `/META-INF` and on your file system in `${was.server.dir}/applicationData/myApplication`:

```
<dir targetInArchive="/META-INF"
 sourceOnDisk="${was.server.dir}/applicationData/myApplication" />
```

To add the directory to an archive so it appears to be in `/apps/jarName.jar/META-INF`, embed the `<dir>` element as follows:

```
<archive targetInArchive="/apps/jarName.jar">
 <dir targetInArchive="/META-INF"
 sourceOnDisk="${was.server.dir}/applicationData/myApplication" />
</archive>
```

In both of the previous examples, all files that are in `${was.server.dir}/applicationData/myApplication` are mapped and visible in the loose application configuration under the directory that is mapped by the `targetInArchive` attribute.

## Virtual paths and file names

If you add `<file>` or `<dir>` elements to an archive, the name of the file or directory in the loose archive does not need to be the same as the actual name on disk.

The following code is an example of how you can configure `${was.server.dir}/applicationFiles/newfile.txt` to appear in the archive as `/application.txt`:

```
<archive>
 <file targetInArchive="/application.txt"
 sourceOnDisk="${was.server.dir}/applicationFiles/newfile.txt"/>
</archive>
```

The same concept also holds true for the path of any added file or directory. The physical resource on disk does not need to be in a directory hierarchy that corresponds to the one being declared.

The following code is an example of how you can make `${was.server.dir}/applicationFiles/newfile.txt` appear in the archive as `/only/available/in/application.txt`:

```
<archive>
 <file targetInArchive="/only/available/in/application.txt"
 sourceOnDisk=" "${was.server.dir}/applicationFiles/newfile.txt"/>
</archive>
```

In each case, the Liberty server sees the resource by the name and path declared by the `targetInArchive` attribute. The Liberty server can navigate the directory hierarchy declared, even if the hierarchy contains only virtual elements, as in the previous example.

```
<archive>
 <file targetInArchive="/only/available/in/red.txt"
 sourceOnDisk="${was.server.dir}/applicationFiles/newfile.txt" />
<archive targetInArchive="/apps/jarName.jar">
 <dir targetInArchive="/META-INF"
 sourceOnDisk="${was.server.dir}/applicationData/myApplication" />
</archive>
```

## Folders and files with the same name

If you have two folders with the same name in the same virtual location in the loose application configuration, the folders are merged, and the contents of both folders are available. If you have two files with the same target location in the loose archive, the first occurrence of the file is used. The first occurrence is based on a top-down approach to reading the elements of the loose application configuration file.

If the first file found is the wrong file, reorder the XML so that the element that contains the version of the file you want is processed first. The first occurrence applies to files defined in `<dir>` elements and files that are defined in `<file>` elements. The first occurrence of a file with the same name and virtual location is the one returned from the virtual file system.

## Considerations for loose applications

For all loose configured applications, the files are not on disk in the hierarchy that they are declared to be. If your applications directly access their own resources, and expect them to be arranged on disk as they would be with an expanded war or ear layout, they might exhibit unexpected behavior.

You can use `ServletContext.getRealPath` in your applications to discover physical resource paths. `ServletContext.getRealPath` can discover file paths to open to read or write data, and obtain directories. However, if you use `ServletContext.getRealPath` in web applications to obtain a path for `"/"`, you cannot use this path to navigate the application on disk.

`ServletContext.getRealPath` allows only a single physical path to be returned, and the loose application might have merged multiple directories to form one path visible to the application.

Consider the following configuration:

```
<archive>
 <dir targetInArchive="/"
 sourceOnDisk="c:\myapplication" />
 <dir targetInArchive="/web/pages"
 sourceOnDisk="c:\webpagesforapplication" />
</archive>
```

An application that directly accesses `/web/pages` and then navigates up the directory hierarchy, finds that the parent of the physical path of `/web/pages` is `c:\` and not `/web`. `c:\` has no `pages` directory and no parent directory.

These considerations apply only if your applications attempt to directly access the content on disk, and perform their own path navigation based on an assumption of a corresponding hierarchical layout on disk. The same applications also encounter issues if they are deployed as an archive. These applications generally experience issues with portability.

## Complex example

The following code is a more complex example of loose application configuration. This example uses all of the elements and creates a complex mapping of files and directories:

```
<archive>
 <dir targetInArchive="/appResources"
 sourceOnDisk="${was.server.dir}/applicationFiles" />
 <archive targetInArchive="application.jar">
 <dir targetInArchive="/src"
 sourceOnDisk="${was.server.dir}/applicationCode/src" />
 </archive>
 <archive targetInArchive="webApp.war">
 <dir targetInArchive="/META-INF"
 sourceOnDisk="${was.server.dir}/manifestFiles/" />
 </archive>
```

```

<dir targetInArchive="/WEB-INF"
 sourceOnDisk="c:/myWorkspace/webAppProject/web-inf" />
<archive targetInArchive="/WEB-INF/lib/myUtility.jar">
 <dir targetInArchive="/"
 sourceOnDisk="c:/myWorkspace/myUtilityProject/src" />
 <file targetInArchive="/someJar.jar"
 sourceOnDisk="c:/myWorkspace/myUtilityProject/aJar.jar" />
</archive>
</archive>
<file targetInArchive="/myjar.jar"
 sourceOnDisk="{was.server.dir}/apps/application.zip" />
</archive>

```

---

## Discovering REST API documentation on a Liberty server

8.5.5.8

You can discover your REST API documentation. Use the API Discovery feature to find what REST APIs are available on a Liberty server and then use the Swagger user interface to invoke the found REST endpoints.

### Procedure

1. Add the `apiDiscovery-1.0` feature to a feature manager in the `server.xml` file of the Liberty server whose available REST APIs you want to find.

The `apiDiscovery-1.0` feature enables the REST API discovery bundles in the product. The feature also exposes documentation from Liberty REST endpoints such as JMX, if the server configuration uses the `restConnector-1.0` feature, and collectives, if the server configuration uses the `collectiveController-1.0` feature.

Ensure the server configuration has all features needed for your deployed application, such as `servlet-3.0`, `jsp-2.2`, and so on. Also ensure the ports and user registry settings are correct for the deployed application.

The following `server.xml` file has the `apiDiscovery-1.0` feature:

```

<server>
 <featureManager>
 <feature>apiDiscovery-1.0</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 httpPort="8010"
 httpsPort="8020"/>

 <keyStore id="defaultKeyStore" password="Liberty"/>

 <basicRegistry id="basic" realm="ibm/api">
 <user name="bob" password="bobpwd" />
 </basicRegistry>
</server>

```

2. Expose the Swagger 2.0 documentation in Liberty REST endpoints.

You can configure the location of your API documentation in either of two ways:

- Use the `getDocument` method of the SPI `com.ibm.wsspi.rest.api.discovery.APIProvider` interface. The `getDocument` method enables OSGi bundles from extension features to contribute REST API documents to the overall “master” documentation. For this release, the only supported `DocType` are `DocType.Swagger_20_JSON` and `DocType.Swagger_20_YAML`. Implementers of this interface can return the serialized JSON or YAML document as a `java.lang.String` value, or they can pass in a file reference (prefixed with `file:///`) to the JSON or YAML file location.
- Use a deployed web application. Each web module can contribute its own REST API document. Multiple WAR files inside an enterprise application (EAR) file can have different Swagger 2.0 documents.



The easiest way to expose the documentation of web modules is to include a `swagger.json` or `swagger.yaml` file inside the corresponding `META-INF` folder. During application deployment, the API Discovery feature looks for a `META-INF/swagger.json` value for each of the web modules. If a `META-INF/swagger.json` value is not found then the API Discovery feature looks for a `META-INF/swagger.yaml` value.

Another way to expose the REST API documentation for a web module is in a `server.xml` configuration file. Put a `webModuleDoc` element for each web module in a parent `apiDiscovery` element; for example:

```
<apiDiscovery>
 <webModuleDoc contextRoot="/30ExampleServletInEar" enabled="true" docURL="/swagger.json" />
 <webModuleDoc contextRoot="/22ExampleServlet" enabled="false" />
 <webModuleDoc contextRoot="/custom" enabled="true" docURL="http://petstore.swagger.io/v2/swagger.json" />
</apiDiscovery>
```

The `webModuleDoc` element must have a `contextRoot` attribute to uniquely identify the web module whose documentation you want to expose.

An optional attribute, `enabled`, toggles the API discovery for a web module. The default of this attribute is `true`.

The `docURL` attribute specifies where to find the documentation for the web module. The `docURL` value can start with a forward slash (/) so that the URL is relative to the context root; for example, `/30ExampleServletInEar/swagger.json`. Or the `docURL` value can start with `http` or `https` for an absolute URL that identifies the complete location of the documentation.

**8.5.5.9** If the web application does not provide a `swagger.json` or `swagger.yaml` file and the application contains JAX-RS annotated resources, you can automatically generate the Swagger document. The server configuration must have the `apiDiscovery-1.0` feature and the `jaxrs-1.1` or `jaxrs-2.0` feature; for example:

```
<featureManager>
 <feature>apiDiscovery-1.0</feature>
 <feature>jaxrs-1.0</feature>
</featureManager>
```

The product scans all classes in the web application for JAX-RS and Swagger annotations, searching for classes with `@Path`, `@Api`, and `@SwaggerDefinition` annotations. The product also automatically generates a corresponding Swagger document during the web application deployment or startup.

**8.5.5.9** You can also use the `apiDiscovery-1.0` feature to merge previously generated documentation with the documentation that it finds during annotation scanning. The product searches for a `META-INF/stub/swagger.json` or `META-INF/stub/swagger.yaml` file in the web module. If the feature finds either of these files, the feature generates a Swagger document that contains both the content of the file and any JAX-RS and Swagger annotations that are in the web module. You can use this feature to document non-JAX-RS servlets because the documentation is automatically merged with the JAX-RS portions.

**8.5.5.9** Another way to expose the REST API documentation for a web module is to add `swagger.json` or `swagger.yaml` to the context root of the web application; for example:

```
http://host:http_port/context_root/swagger.json
```

or

```
http://host:http_port/context_root/swagger.yaml
```

A call to `context_root/swagger.json` or `context_root/swagger.yaml` returns the documentation for the context root in the requested JSON or YAML format. During web module startup, the API Discovery feature pulls any available `context_root/swagger.json` into the aggregated document from `/ibm/api/docs` and `/ibm/api/explorer`. If `context_root/swagger.json` is not available, then the API Discovery feature pulls any available `context_root/swagger.yaml` into the aggregated document from `/ibm/api/docs` and `/ibm/api/explorer`. The `apiDiscovery-1.0` feature can handle `swagger.json` or `swagger.yaml` virtually, which means that if the web application has a `server.xml` file that configures a `docURL` attribute, a `swagger.json` or `swagger.yaml` file in the `META-INF` folder,

or JAX-RS and Swagger annotations, then the call to `context_root/swagger.json` or `context_root/swagger.yaml` returns the documentation for this other configuration in the requested JSON or YAML format.

### 3. Discover your API documentation.

After you configure the location of your API documentation, you can discover it in the following ways:

- Use the GET `https://host:https_port/ibm/api/docs` endpoint.

This endpoint provides a valid Swagger 2.0 document with all available Liberty REST APIs merged into a single document. This is useful for consumer applications that want to programmatically navigate the set of available APIs, such as an API Management solution. Including an optional Accept header with an `application/yaml` value provides the Swagger 2.0 document in YAML format. This endpoint has a multiple-cardinality, optional query parameter called `root` that can filter the found context roots. For example, a call to GET `https://host:https_port/ibm/api/docs?root=/myApp` retrieves a Swagger 2.0 document that only has the documentation for the `myApp` context root.

- Use the GET `https://host:https_port/ibm/api/explorer` endpoint.

This endpoint provides an attractive, rendered HTML page that displays the content from the `/ibm/api/docs` URL. This page follows the same pattern as the standard online sample (<http://petstore.swagger.io/>), so that users can invoke the REST API. This endpoint helps you explore the available REST APIs on a Liberty server, and perhaps invoke them from the page. A filter input box enables a comma-separated list of context roots to filter the content. This filtering works like the `root` query parameter. You can test drive the APIs by providing the required input values and click the "Try it out" button.

**8.5.5.9** After you enable the `apiDiscovery-1.0` feature, the management bean (MBean) that has the `ObjectName` value `WebSphere:feature=apiDiscovery,name=APIDiscovery` is registered in the Liberty MBeanServer. This MBean provides the following attributes:

- The `DocumentationURL` attribute is the full URL of the `/ibm/api/docs` endpoint, and it displays the raw JSON or YAML documentation.
- The `ExplorerURL` attribute is the full URL of the `/ibm/api/explorer` endpoint, and it displays the rendered UI of the documentation.

You can use this MBean to learn whether REST API Discovery is enabled and where a client can reach it.

## Subscribe to Liberty REST API updates

**8.5.5.9**

The Liberty REST API discovery feature now exposes a new REST API, `/ibm/api/docs/subscription`, which allows users to subscribe to any REST API update, such as new APIs being available or old APIs being removed. This is useful when a user wants to be notified immediately of any changes in the endpoints that are provided by a particular Liberty instance.

- Enable subscriptions
- Example request and response

### Enable subscriptions

In addition to the base `apiDiscovery-1.0` configuration, it is required to also configure either `websocket-1.0` or `websocket-1.1` in your `server.xml`.

```
<server>
 <featureManager>
 <feature>apiDiscovery-1.0</feature>
 <feature>websocket-1.1</feature>
 </featureManager>

 <httpEndpoint id="defaultHttpEndpoint"
 host="*"
 >
```

```

 httpPort="8010"
 httpsPort="8020"/>

 <keyStore id="defaultKeyStore" password="Liberty"/>

 <basicRegistry id="basic" realm="ibm/api">
 <user name="bob" password="bobpwd" />
 </basicRegistry>
</server>

```

The `/ibm/api/docs/subscription` endpoint allows for POST requests with a JSON payload in the format:

```
{ "docType" : String }
```

Where the String can be one of `Swagger_20_JSON` or `Swagger_20_YAML`. The returning JSON payload outlines the type of subscription feed and its URL.

## Example request and corresponding response

Request:

```
{"docType": "Swagger_20_JSON"}
```

Response:

```
{
 "feedType": "websocket",
 "feedURL": "wss://myserver.com:8020/ibm/api/docs/subscription/websocket/60db0d79-1863-48f5-a0f9-4fe22a27b82d"
}
```

You can now use a websocket client to connect to the feed URL. Once connected, any further updates to REST APIs in the Liberty server is pushed through the websocket. The update is either in JSON or YAML format, depending on the subscription.

## REST endpoints for pushing APIs into IBM API Connect

8.5.5.9

Use the REST endpoint, which is a central location for both on-premises and cloud Liberty users, to visualize, call, and push APIs into IBM API Connect.

### Pushing deployed REST endpoints into IBM API Connect

To push deployed REST endpoints into IBM API Connect, you must call a new REST endpoint, `/ibm/api/docs/apiconnect`, which is exposed by the `apiDiscovery-1.0` feature in the server configuration. Administrators and developers can use the REST endpoints to expose assets from a Liberty instance to any enterprise developer who is searching the IBM API Connect framework catalogs.

### Providing a product definition

All APIs are referenced by a product and exposed from a catalog. Therefore, the caller provides a product definition that Liberty uses to refer to its RESTful APIs and push the resulting product into IBM API Connect. A sample product is provided in this topic.

### Exposing assets of a Liberty collective into IBM API Connect

Using a corresponding Liberty collective endpoint, `/ibm/api/collective/docs/apiconnect`, you can expose all assets of a Liberty Collective into IBM API Connect with a single RESTful trigger. The Liberty collective endpoint can expose thousands of APIs to any cloud developer connected to API Connect. See the Liberty RESTful API registry, `/ibm/api/explorer`, for the full Swagger definition of this endpoint.

## Endpoint summary

HTTP Request method: POST

URL: `https://server:https_port/ibm/api/docs/apiconnect`

## Required headers

### X-APIM-Authorization

Credentials to connect to API Connect display in the two following forms:

- username and password
- xyz where xyz is the base64 encoded version of username: password.

## Required query parameters

**Server** The name of the IBM API Connect server, starting with `https://`.

### Catalog

The name of the catalog that hosts the resulting product.

### Organization

The name of the organization of the caller.

## Optional query parameters

### apiRoot

A multi-cardinality parameter that specifies exactly which context roots, such as, `apiRoot=/myApp`, the caller wants to push into API Connect. By default, Liberty includes any deployed application except known Liberty runtime Web Application Bundles. This parameter is useful when you want to filter which applications get exposed.

### member ID (only available for the Collective variant)

A multi-cardinality parameter that specifies the ID of the exact Collective Members from which the caller wants to expose assets. This ID is composed of a string with the host name, the URLEncoded user dir, and the server name, all separated by a comma, for example: `myHost.com,%2Ftmp%2Fwlp%2Fusr, server1`.

## Input body

Product definition in either in YAML or JSON code. See the following YAML example:

```
product: "1.0.0"
 info:
 name: "pushed-product"
 title: "A Product that encapsulates Liberty APIs"
 version: "1.0.0"
 visibility:
 view:
 enabled: true
 type: "public"
 tags:
 - "string"
 orgs:
 - "string"
 subscribe:
 enabled: true
 type: "authenticated"
 tags:
 - "string"
 orgs:
 - "string"
 apis:
 liberty:
 name: "liberty-api:1.0.0"
 x-ibm-configuration:
 phase: "realized"
 testable: true
 enforced: true
```

```
cors:
 enabled: true
assembly:
 execute:
 -
 invoke:
 target-url: "${gateway.target}"
 title: "Invocation"
 description: "Invoking back-end service"
plans:
 default:
 title: "Default Plan"
 rate-limit:
 hard-limit: false
 value: "100/hour"
 approval: false
 createdAt: "2016-04-18T20:33:22.937Z"
 createdBy: "string"
```



---

## Chapter 10. Monitoring the Liberty server runtime environment

You can use the `monitor-1.0` feature to monitor the server runtime environment.

### About this task

To enable monitoring for your Liberty server, you add `monitor-1.0` Liberty feature in the `server.xml` file.

The `monitor-1.0` feature provides monitoring support for user runtime components.

- JVM
- Web applications
- Thread pool
- Session management
- ConnectionPool

For more details, see “Liberty features” on page 483.

### Procedure

1. Add the `monitor-1.0` feature and the monitoring starts.

**Note:** 8.5.5.3 If you launch the server by not using the server script (`server.sh` or `server.bat`) on a Java Virtual Machine (JVM), ensure that the JavaAgent is configured for JVM as given in the following example: `agentlib=-javaagent:<path to liberty install>/bin/tools/ws-javaagent.jar`.

2. Monitoring data is reported as standard MXBeans.
3. You can use JConsole to connect to JVM and look at the performance data by clicking each attribute of the MXBean.

The MXBeans for monitoring are as following:

- `WebSphere:type=JvmStats`
  - `WebSphere:type=ServletStats,name=*`
  - `WebSphere:type=ThreadPoolStats,name=Default Executor`
  - `WebSphere:type=SessionStats,name=*`
  - `WebSphere:type=ConnectionPool,name=*`
4. Optional: The same data is available with traditional PMI MBean (Perf MBean). Note that the Perf MBean is stabilized.

---

### JVM monitoring

You can use the `JvmStats` MXBean for JVM monitoring in Liberty.

Each Liberty instance has one `JvmStats` MXBean.

The `ObjectName` for identifying JVM MXBean is:

```
WebSphere:type=JvmStats
```

Available Instances = 1

This MXBean is responsible for reporting performance of JVM. Following attributes are available for JVM.

### Heap Information

- Amount of free heap available (in Bytes)
- Total used memory by JVM for from heap (in Bytes)
- Heap size (in Bytes)

### CPU Information

- Percentage of CPU consumed by this JVM

### Garbage Collection (GC) Information

- Number of times that GC happened since JVM started
- Total time taken by GC activity

### General Information

- Time in milliseconds since JVM has started.

### Counter definitions (Attributes to MXBean)

- Heap: Heap size used for current JVM.
- FreeMemory: Free heap available for current JVM.
- UsedMemory: Used heap for current JVM.
- ProcessCPU: Percentage of CPU used by JVM process.
- GcCount: Number of times GC has happened since JVM starts.
- GcTime: Total accumulated value of GC time.
- UpTime: Time in milliseconds, since JVM has started.

### Management Interface

The management interface of JVM monitoring is `com.ibm.websphere.monitor.jmx.JvmMXBean`. You can use the management interface to obtain a proxy object. See “Examples of accessing MBean attributes and operations” on page 1029.

For more information about the management interface, see the Java API document for Liberty. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

---

## Web application monitoring

You can use a servlet MXBean for web application monitoring of Liberty.

Performance data is available for each Servlet in the Web Application. Each servlet has its own MXBean.

The ObjectName for identifying each servlet MXBean is:

```
WebSphere:type=ServletStats,name=<AppName>.<ServletName>
```

For example:

```
WebSphere:type=ServletStats,name=snoop.Alpine Snoop Servlet
WebSphere:type=ServletStats,name=MyApp.MyServlet
```



This MBean is responsible for reporting ServletStats for each servlet. Following key data is available for ServletStats MBean:

- Request Count
- Response Time
- Servlet Name
- Application Name

#### Counter definitions (Attributes to MBean)

- `AppName`: Name of the application.
- `ServletName`: Name of the servlet.
- `RequestCount`: Number of hits to this servlet.
- `ResponseTime`: Average response time (nano-seconds).
- `Description`: Description of counter.
- `RequestCountDetails`: RequestCount details including last time stamp.
- `ResponseTimeDetails`: ResponseTime details including number of snapshot taken, min, and max values.

#### Management Interface

The management interface of web application monitoring is `com.ibm.websphere.webcontainer.ServletStatsMBean`. You can use the management interface to obtain a proxy object. See “Examples of accessing MBean attributes and operations” on page 1029.

For more information about the management interface, see the Java API document for Liberty. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

---

## ThreadPool monitoring

You can use a ThreadPool MBean for thread pool monitoring in Liberty.

All Web Requests executes in the thread pool, named **Default Executor** thread pool. You can monitor the usage of **Default Executor** thread pool using ThreadPoolMBean.

The ObjectName for identifying MBean for thread pool is :

```
WebSphere:type=ThreadPoolStats,name=Default Executor
```

Key Performance data available for ThreadPool are:

- Threads in the pool which represents the pool size.
- Active threads which are serving requests.

#### Attributes for ThreadPool

- `ActiveThreads`
- `PoolSize`
- `PoolName` (Only supports Default Executor thread pool)

#### Management Interface

The management interface of ThreadPool monitoring is

`com.ibm.websphere.monitor.jmx.ThreadPoolMBean`. You can use the management interface to obtain a proxy object. See “Examples of accessing MBean attributes and operations” on page 1029.

For more information about the management interface, see the Java API document for Liberty. The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

## SIP application monitoring

8.5.5.9

Session Initiation Protocol (SIP) Performance Monitoring Infrastructure (PMI) is a component that collects SIP performance metrics of a running application server. To monitor the SIP metrics, you must enable PMI on your server. To enable monitoring for SIP, add `monitor-1.0` and `sipServlet-1.1` Liberty features to the `server.xml` file.

All Liberty SIP PMI counters are shown by standard MBeans.

The SIP container provides the following MBean interfaces for the SIP counters:

- `WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic`
- `WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration`
- `WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest`
- `WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest`
- `WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse`
- `WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse`
- `WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor`

Each interface shows a different set of SIP PMI metrics. Refer to the tables for module details. You can view the SIP PMI counters in multiple ways:

- Use JConsole to connect to JVM and look at the SIP PMI counters by clicking each attribute of the MBean.
- Create your own JMX client application to inspect the counters by starting the MBean operations.

For more information about how to create a JMX client to start the MBean operations, see [Connecting to Liberty by using JMX and Liberty: Examples of accessing MBean attributes and operations](#). SIP provides the following counters in PMI to monitor SIP performance.

*Table 97. SIP container basic counters.* The object name of the MBean from which the counters can be retrieved is: `"WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic"`. To retrieve the attributes, use the `JMXConnection.getAttribute` method. For example: `_connection.getAttribute("WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic", "SipAppSessions")`.

This table lists the SIP container basic counters.

Name	Attribute	Description	Granularity
Incoming traffic	ReceivedSipMsgs	The average number of messages that are handled by the container and calculated over a configurable period	Server

*Table 97. SIP container basic counters (continued).* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic". To retrieve the attributes, use the JMXConnection.getAttribute method. For example: `_connection.getAttribute("WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic","SipAppSessions")`.

This table lists the SIP container basic counters.

Name	Attribute	Description	Granularity
New SIP application sessions	NewSipApplications	The average number of new SIP application sessions created in the container and calculated over a configurable period	Server
Response time	SipRequestProcessing	The average amount of time that it takes between when a message gets into the container and when a response is sent from the container	Server
Queue size	InvokerSize	The size of the invoke queue in the product	Server
Rejected SIP messages	RejectedMessages	The number of rejected SIP messages	Server
SIP timer invocations	SipTimersInvocations	The number of invocations of the SIP timers (Timer A, Timer B, Timer C, Timer D, Timer E, Timer F, Timer G, Timer H)	Server
Number of active SIP sessions	SipSessions	The number of SIP sessions that belong to each application	Server
Number of active SIP application sessions	SipAppSessions	The number of SIP application sessions that belong to each application	Server

*Table 98. SIP container inbound requests.* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest","getTotalInboundRequests",_appName,"INVITE")`.

This table lists the inbound request counters.

Name	Method	Description	Granularity
Number of inbound NOT SIP STANDARD requests	<code>getTotalInboundRequests(appName,"NOTSIPSTANDARD");</code>	Number of inbound NOT SIP STANDARD requests that belong to each application	Application
Number of inbound REGISTER requests	<code>getTotalInboundRequests(appName,"REGISTER");</code>	Number of inbound REGISTER requests that belong to each application	Application
Number of inbound INVITE requests	<code>getTotalInboundRequests(appName,"INVITE");</code>	Number of inbound INVITE requests that belong to each application	Application

*Table 98. SIP container inbound requests (continued).* The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest", "getTotalInboundRequests", _appName, "INVITE")`.

This table lists the inbound request counters.

Name	Method	Description	Granularity
Number of inbound ACK requests	<code>getTotalInboundRequests(applicationName, "ACK");</code>	Number of inbound ACK requests that belong to each application	Application
Number of inbound OPTIONS requests	<code>getTotalInboundRequests(applicationName, "OPTIONS");</code>	Number of inbound OPTIONS requests that belong to each application	Application
Number of inbound BYE requests	<code>getTotalInboundRequests(applicationName, "BYE");</code>	Number of inbound BYE requests that belong to each application	Application
Number of inbound CANCEL requests	<code>getTotalInboundRequests(applicationName, "CANCEL");</code>	Number of inbound CANCEL requests that belong to each application	Application
Number of inbound PRACK requests	<code>getTotalInboundRequests(applicationName, "PRACK");</code>	Number of inbound PRACK requests that belong to each application	Application
Number of inbound INFO requests	<code>getTotalInboundRequests(applicationName, "INFO");</code>	Number of inbound INFO requests that belong to each application	Application
Number of inbound SUBSCRIBE requests	<code>getTotalInboundRequests(applicationName, "SUBSCRIBE");</code>	Number of inbound SUBSCRIBE requests that belong to each application	Application
Number of inbound NOTIFY requests	<code>getTotalInboundRequests(applicationName, "NOTIFY");</code>	Number of inbound NOTIFY requests that belong to each application	Application
Number of inbound MESSAGE requests	<code>getTotalInboundRequests(applicationName, "MESSAGE");</code>	Number of inbound MESSAGE requests that belong to each application	Application
Number of inbound PUBLISH requests	<code>getTotalInboundRequests(applicationName, "PUBLISH");</code>	Number of inbound PUBLISH requests that belong to each application	Application
Number of inbound REFER requests	<code>getTotalInboundRequests(applicationName, "REFER");</code>	Number of inbound REFER requests that belong to each application	Application
Number of inbound UPDATE requests	<code>getTotalInboundRequests(applicationName, "UPDATE");</code>	Number of inbound UPDATE requests that belong to each application	Application

*Table 99. SIP container inbound responses.* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100")`.

This table lists the inbound response counters.

Name	Method	Description	Granularity
Number of inbound 100 responses	<code>getTotalInboundResponses(applicationName, "100");</code>	Number of inbound 100 (Trying) responses that belong to each application	Application
Number of inbound 180 responses	<code>getTotalInboundResponses(applicationName, "180");</code>	Number of inbound 180 (Ringing) responses that belong to each application	Application
Number of inbound 181 responses	<code>getTotalInboundResponses(applicationName, "181");</code>	Number of inbound 181 (Call being forwarded) responses that belong to each application	Application
Number of inbound 182 responses	<code>getTotalInboundResponses(applicationName, "182");</code>	Number of inbound 182 (Call Queued) responses that belong to each application	Application
Number of inbound 183 responses	<code>getTotalInboundResponses(applicationName, "183");</code>	Number of inbound 183 (Session Progress) responses that belong to each application	Application
Number of inbound 200 responses	<code>getTotalInboundResponses(applicationName, "200");</code>	Number of inbound 200 (OK) responses that belong to each application	Application
Number of inbound 202 responses	<code>getTotalInboundResponses(applicationName, "202");</code>	Number of inbound 202 (Accepted) responses that belong to each application	Application
Number of inbound 300 responses	<code>getTotalInboundResponses(applicationName, "300");</code>	Number of inbound 300 (Multiple choices) responses that belong to each application	Application
Number of inbound 301 responses	<code>getTotalInboundResponses(applicationName, "301");</code>	Number of inbound 301 (Moved Permanently) responses that belong to each application	Application
Number of inbound 302 responses	<code>getTotalInboundResponses(applicationName, "302");</code>	Number of inbound 302 (Moved Temporarily) responses that belong to each application	Application
Number of inbound 305 responses	<code>getTotalInboundResponses(applicationName, "305");</code>	Number of inbound 305 (Use Proxy) responses that belong to each application	Application
Number of inbound 380 responses	<code>getTotalInboundResponses(applicationName, "380");</code>	Number of inbound 380 (Alternative Service) responses that belong to each application	Application

Table 99. SIP container inbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100")`.

This table lists the inbound response counters.

Name	Method	Description	Granularity
Number of inbound 400 responses	<code>getTotalInboundResponses(applicationName, "400");</code>	The number of inbound 400 (Bad Request) responses that belong to each application	Application
Number of inbound 401 responses	<code>getTotalInboundResponses(applicationName, "401");</code>	The number of inbound 401 (Unauthorized) responses that belong to each application	Application
Number of inbound 402 responses	<code>getTotalInboundResponses(applicationName, "402");</code>	The number of inbound 402 (Payment Required) responses that belong to each application	Application
Number of inbound 403 responses	<code>getTotalInboundResponses(applicationName, "403");</code>	The number of inbound 403 (Forbidden) responses that belong to each application	Application
Number of inbound 404 responses	<code>getTotalInboundResponses(applicationName, "404");</code>	The number of inbound 404 (Not Found) responses that belong to each application	Application
Number of inbound 405 responses	<code>getTotalInboundResponses(applicationName, "405");</code>	The number of inbound 405 (Method Not Allowed) responses that belong to each application	Application
Number of inbound 406 responses	<code>getTotalInboundResponses(applicationName, "406");</code>	The number of inbound 406 (Not Acceptable) responses that belong to each application	Application
Number of inbound 407 responses	<code>getTotalInboundResponses(applicationName, "407");</code>	The number of inbound 407 (Proxy Authentication Required) responses that belong to each application	Application
Number of inbound 408 responses	<code>getTotalInboundResponses(applicationName, "408");</code>	The number of inbound 408 (Request Timeout) responses that belong to each application	Application
Number of inbound 410 responses	<code>getTotalInboundResponses(applicationName, "410");</code>	The number of inbound 410 (Gone) responses that belong to each application	Application
Number of inbound 413 responses	<code>getTotalInboundResponses(applicationName, "413");</code>	The number of inbound 413 (Request Entity Too Large) responses that belong to each application	Application
Number of inbound 414 responses	<code>getTotalInboundResponses(applicationName, "414");</code>	The number of inbound 414 (Request URI Too Long) responses that belong to each application	Application

Table 99. SIP container inbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100")`.

This table lists the inbound response counters.

Name	Method	Description	Granularity
Number of inbound 415 responses	<code>getTotalInboundResponses(applicationName, "415");</code>	Number of inbound 415 (Unsupported Media Type) responses that belong to each application	Application
Number of inbound 416 responses	<code>getTotalInboundResponses(applicationName, "416");</code>	Number of inbound 416 (Unsupported URI Scheme) responses that belong to each application	Application
Number of inbound 420 responses	<code>getTotalInboundResponses(applicationName, "420");</code>	Number of inbound 420 (Bad Extension) responses that belong to each application	Application
Number of inbound 421 responses	<code>getTotalInboundResponses(applicationName, "421");</code>	Number of inbound 421 (Extension Required) responses that belong to each application	Application
Number of inbound 423 responses	<code>getTotalInboundResponses(applicationName, "423");</code>	Number of inbound 423 (Interval Too Brief) responses that belong to each application	Application
Number of inbound 480 responses	<code>getTotalInboundResponses(applicationName, "480");</code>	Number of inbound 480 (Temporarily Unavailable) responses that belong to each application	Application
Number of inbound 481 responses	<code>getTotalInboundResponses(applicationName, "481");</code>	Number of inbound 481 (Call Leg Done) responses that belong to each application	Application
Number of inbound 482 responses	<code>getTotalInboundResponses(applicationName, "482");</code>	Number of inbound 482 (Loop Detected) responses that belong to each application	Application
Number of inbound 483 responses	<code>getTotalInboundResponses(applicationName, "483");</code>	Number of inbound 483 (Too Many Hops) responses that belong to each application	Application
Number of inbound 484 responses	<code>getTotalInboundResponses(applicationName, "484");</code>	Number of inbound 484 (Address Incomplete) responses that belong to each application	Application
Number of inbound 485 responses	<code>getTotalInboundResponses(applicationName, "485");</code>	Number of inbound 485 (Ambiguous) responses that belong to each application	Application
Number of inbound 486 responses	<code>getTotalInboundResponses(applicationName, "486");</code>	Number of inbound 486 (Busy Here) responses that belong to each application	Application

Table 99. SIP container inbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100")`.

This table lists the inbound response counters.

Name	Method	Description	Granularity
Number of inbound 487 responses	<code>getTotalInboundResponses(applicationName, "487");</code>	Number of inbound 487 (Request Terminated) responses that belong to each application	Application
Number of inbound 488 responses	<code>getTotalInboundResponses(applicationName, "488");</code>	Number of inbound 488 (Not Acceptable Here) responses that belong to each application	Application
Number of inbound 491 responses	<code>getTotalInboundResponses(applicationName, "491");</code>	Number of inbound 491 (Request Pending) responses that belong to each application	Application
Number of inbound 493 responses	<code>getTotalInboundResponses(applicationName, "493");</code>	Number of inbound 493 (Undecipherable) responses that belong to each application	Application
Number of inbound 500 responses	<code>getTotalInboundResponses(applicationName, "500");</code>	Number of inbound 500 (Server Internal Error) responses that belong to each application	Application
Number of inbound 501 responses	<code>getTotalInboundResponses(applicationName, "501");</code>	Number of inbound 501 (Not Implemented) responses that belong to each application	Application
Number of inbound 502 responses	<code>getTotalInboundResponses(applicationName, "502");</code>	Number of inbound 502 (Bad Gateway) responses that belong to each application	Application
Number of inbound 503 responses	<code>getTotalInboundResponses(applicationName, "503");</code>	Number of inbound 503 (Service Unavailable) responses that belong to each application	Application
Number of inbound 504 responses	<code>getTotalInboundResponses(applicationName, "504");</code>	Number of inbound 504 (Server Timeout) responses that belong to each application	Application
Number of inbound 505 responses	<code>getTotalInboundResponses(applicationName, "505");</code>	Number of inbound 505 (Version Not Supported) responses that belong to each application	Application
Number of inbound 513 responses	<code>getTotalInboundResponses(applicationName, "513");</code>	Number of inbound 513 (Message Too Large) responses that belong to each application	Application



*Table 99. SIP container inbound responses (continued).* The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100").`

This table lists the inbound response counters.

Name	Method	Description	Granularity
Number of inbound 600 responses	<code>getTotalInboundResponses(applicationName, "600");</code>	The number of inbound 600 (Busy Everywhere) responses that belong to each application	Application
Number of inbound 603 responses	<code>getTotalInboundResponses(applicationName, "603");</code>	The number of inbound 603 (Decline) responses that belong to each application	Application
Number of inbound 604 responses	<code>getTotalInboundResponses(applicationName, "604");</code>	The number of inbound 604 (Does Not Exit Anywhere) responses that belong to each application	Application
Number of inbound 606 responses	<code>getTotalInboundResponses(applicationName, "606");</code>	The number of inbound 606 (Not Acceptable Anywhere) responses that belong to each application	Application

*Table 100. SIP container outbound requests.* The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest", "getTotalOutboundRequests", _appName, "INVITE").`

This table lists the outbound request counters.

Name	Method	Description	Granularity
Number of outbound NOT SIP STANDARD requests	<code>getTotalOutboundRequests(applicationName, "NOTSIPSTANDARD");</code>	The number of outbound NOT SIP STANDARD requests that belong to each application	Application
Number of outbound REGISTER requests	<code>getTotalOutboundRequests(applicationName, "REGISTER");</code>	The number of outbound REGISTER requests that belong to each application	Application
Number of outbound INVITE requests	<code>getTotalOutboundRequests(applicationName, "INVITE");</code>	The number of outbound INVITE requests that belong to each application	Application
Number of outbound ACK requests	<code>getTotalOutboundRequests(applicationName, "ACK");</code>	The number of outbound ACK requests that belong to each application	Application
Number of outbound OPTIONS requests	<code>getTotalOutboundRequests(applicationName, "OPTIONS");</code>	The number of outbound OPTIONS requests that belong to each application	Application
Number of outbound BYE requests	<code>getTotalOutboundRequests(applicationName, "BYE");</code>	The number of outbound BYE requests that belong to each application	Application

*Table 100. SIP container outbound requests (continued).* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest", "getTotalOutboundRequests", _appName, "INVITE").`

This table lists the outbound request counters.

Name	Method	Description	Granularity
Number of outbound CANCEL requests	<code>getTotalOutboundRequests(applicationName, "CANCEL");</code>	The number of outbound CANCEL requests that belong to each application	Application
Number of outbound PRACK requests	<code>getTotalOutboundRequests(applicationName, "PRACK");</code>	The number of outbound PRACK requests that belong to each application	Application
Number of outbound INFO requests	<code>getTotalOutboundRequests(applicationName, "INFO");</code>	The number of outbound INFO requests that belong to each application	Application
Number of outbound SUBSCRIBE requests	<code>getTotalOutboundRequests(applicationName, "SUBSCRIBE");</code>	The number of outbound SUBSCRIBE requests that belong to each application	Application
Number of outbound NOTIFY requests	<code>getTotalOutboundRequests(applicationName, "NOTIFY");</code>	The number of outbound NOTIFY requests that belong to each application	Application
Number of outbound MESSAGE requests	<code>getTotalOutboundRequests(applicationName, "MESSAGE");</code>	The number of outbound MESSAGE requests that belong to each application	Application
Number of outbound PUBLISH requests	<code>getTotalOutboundRequests(applicationName, "PUBLISH");</code>	The number of outbound PUBLISH requests that belong to each application	Application
Number of outbound REFER requests	<code>getTotalOutboundRequests(applicationName, "REFER");</code>	The number of outbound REFER requests that belong to each application	Application
Number of outbound UPDATE requests	<code>getTotalOutboundRequests(applicationName, "UPDATE");</code>	The number of outbound UPDATE requests that belong to each application	Application

*Table 101. SIP container outbound responses.* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100").`

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 100 responses	<code>getTotalOutboundResponses(applicationName, "100");</code>	The number of outbound 100 (Trying) responses that belong to each application	Application
Number of outbound 180 responses	<code>getTotalOutboundResponses(applicationName, "180");</code>	The number of outbound 180 (Ringing) responses that belong to each application	Application

*Table 101. SIP container outbound responses (continued).* The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100").`

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 181 responses	<code>getTotalOutboundResponses("181");</code>	Number of outbound 181 (Call being forwarded) responses that belong to each application	Application
Number of outbound 182 responses	<code>getTotalOutboundResponses("182");</code>	Number of outbound 182 (Call Queued) responses that belong to each application	Application
Number of outbound 183 responses	<code>getTotalOutboundResponses("183");</code>	Number of outbound 183 (Session Progress) responses that belong to each application	Application
Number of outbound 200 responses	<code>getTotalOutboundResponses("200");</code>	Number of outbound 200 (OK) responses that belong to each application	Application
Number of outbound 202 responses	<code>getTotalOutboundResponses("202");</code>	Number of outbound 202 (Accepted) responses that belong to each application	Application
Number of outbound 300 responses	<code>getTotalOutboundResponses("300");</code>	Number of outbound 300 (Multiple choices) responses that belong to each application	Application
Number of outbound 301 responses	<code>getTotalOutboundResponses("301");</code>	Number of outbound 301 (Moved Permanently) responses that belong to each application	Application
Number of outbound 302 responses	<code>getTotalOutboundResponses("302");</code>	Number of outbound 302 (Moved Temporarily) responses that belong to each application	Application
Number of outbound 305 responses	<code>getTotalOutboundResponses("305");</code>	Number of outbound 305 (Use Proxy) responses that belong to each application	Application
Number of outbound 380 responses	<code>getTotalOutboundResponses("380");</code>	Number of outbound 380 (Alternative Service) responses that belong to each application	Application
Number of outbound 400 responses	<code>getTotalOutboundResponses("400");</code>	Number of outbound 400 (Bad Request) responses that belong to each application	Application

Table 101. SIP container outbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100")`.

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 401 responses	<code>getTotalOutboundResponses("401");</code>	Number of outbound 401 (Unauthorized) responses that belong to each application	Application
Number of outbound 402 responses	<code>getTotalOutboundResponses("402");</code>	Number of outbound 402 (Payment Required) responses that belong to each application	Application
Number of outbound 403 responses	<code>getTotalOutboundResponses("403");</code>	Number of outbound 403 (Forbidden) responses that belong to each application	Application
Number of outbound 404 responses	<code>getTotalOutboundResponses("404");</code>	Number of outbound 404 (Not Found) responses that belong to each application	Application
Number of outbound 405 responses	<code>getTotalOutboundResponses("405");</code>	Number of outbound 405 (Method Not Allowed) responses that belong to each application	Application
Number of outbound 406 responses	<code>getTotalOutboundResponses("406");</code>	Number of outbound 406 (Not Acceptable) responses that belong to each application	Application
Number of outbound 407 responses	<code>getTotalOutboundResponses("407");</code>	Number of outbound 407 (Proxy Authentication Required) responses that belong to each application	Application
Number of outbound 408 responses	<code>getTotalOutboundResponses("408");</code>	Number of outbound 408 (Request Timeout) responses that belong to each application	Application
Number of outbound 410 responses	<code>getTotalOutboundResponses("410");</code>	Number of outbound 410 (Gone) responses that belong to each application	Application
Number of outbound 413 responses	<code>getTotalOutboundResponses("413");</code>	Number of outbound 413 (Request Entity Too Large) responses that belong to each application	Application
Number of outbound 414 responses	<code>getTotalOutboundResponses("414");</code>	Number of outbound 414 (Request URI Too Long) responses that belong to each application	Application

Table 101. SIP container outbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100")`.

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 415 responses	<code>getTotalOutboundResponses("415");</code>	Number of outbound 415 (Unsupported Media Type) responses that belong to each application	Application
Number of outbound 416 responses	<code>getTotalOutboundResponses("416");</code>	Number of outbound 416 (Unsupported URI Scheme) responses that belong to each application	Application
Number of outbound 420 responses	<code>getTotalOutboundResponses("420");</code>	Number of outbound 420 (Bad Extension) responses that belong to each application	Application
Number of outbound 421 responses	<code>getTotalOutboundResponses("421");</code>	Number of outbound 421 (Extension Required) responses that belong to each application	Application
Number of outbound 423 responses	<code>getTotalOutboundResponses("423");</code>	Number of outbound 423 (Interval Too Brief) responses that belong to each application	Application
Number of outbound 480 responses	<code>getTotalOutboundResponses("480");</code>	Number of outbound 480 (Temporarily Unavailable) responses that belong to each application	Application
Number of outbound 481 responses	<code>getTotalOutboundResponses("481");</code>	Number of outbound 481 (Call Leg Done) responses that belong to each application	Application
Number of outbound 482 responses	<code>getTotalOutboundResponses("482");</code>	Number of outbound 482 (Loop Detected) responses that belong to each application	Application
Number of outbound 483 responses	<code>getTotalOutboundResponses("483");</code>	Number of outbound 483 (Too Many Hops) responses that belong to each application	Application
Number of outbound 484 responses	<code>getTotalOutboundResponses("484");</code>	Number of outbound 484 (Address Incomplete) responses that belong to each application	Application
Number of outbound 485 responses	<code>getTotalOutboundResponses("485");</code>	Number of outbound 485 (Ambiguous) responses that belong to each application	Application

Table 101. SIP container outbound responses (continued). The object name of the MXBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100")`.

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 486 responses	<code>getTotalOutboundResponses("486");</code>	Number of outbound 486 (Busy Here) responses that belong to each application	Application
Number of outbound 487 responses	<code>getTotalOutboundResponses("487");</code>	Number of outbound 487 (Request Terminated) responses that belong to each application	Application
Number of outbound 488 responses	<code>getTotalOutboundResponses("488");</code>	Number of outbound 488 (Not Acceptable Here) responses that belong to each application	Application
Number of outbound 491 responses	<code>getTotalOutboundResponses("491");</code>	Number of outbound 491 (Request Pending) responses that belong to each application	Application
Number of outbound 493 responses	<code>getTotalOutboundResponses("493");</code>	Number of outbound 493 (Undecipherable) responses that belong to each application	Application
Number of outbound 500 responses	<code>getTotalOutboundResponses("500");</code>	Number of outbound 500 (Server Internal Error) responses that belong to each application	Application
Number of outbound 501 responses	<code>getTotalOutboundResponses("501");</code>	Number of outbound 501 (Not Implemented) responses that belong to each application	Application
Number of outbound 502 responses	<code>getTotalOutboundResponses("502");</code>	Number of outbound 502 (Bad Gateway) responses that belong to each application	Application
Number of outbound 503 responses	<code>getTotalOutboundResponses("503");</code>	Number of outbound 503 (Service Unavailable) responses that belong to each application	Application
Number of outbound 504 responses	<code>getTotalOutboundResponses("504");</code>	Number of outbound 504 (Server Timeout) responses that belong to each application	Application
Number of outbound 505 responses	<code>getTotalOutboundResponses("505");</code>	Number of outbound 505 (Version Not Supported) responses that belong to each application	Application

*Table 101. SIP container outbound responses (continued).* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse". To retrieve the counters, use the JMXConnection.invoke method. For example, `_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100")`.

This table lists the outbound response counters.

Name	Method	Description	Granularity
Number of outbound 513 responses	<code>getTotalOutboundResponses(applicationName, "513");</code>	The number of outbound 513 (Message Too Large) responses that belong to each application	Application
Number of outbound 600 responses	<code>getTotalOutboundResponses(applicationName, "600");</code>	The number of outbound 600 (Busy Everywhere) responses that belong to each application	Application
Number of outbound 603 responses	<code>getTotalOutboundResponses(applicationName, "603");</code>	The number of outbound 603 (Decline) responses that belong to each application	Application
Number of outbound 604 responses	<code>getTotalOutboundResponses(applicationName, "604");</code>	The number of outbound 604 (Does Not Exit Anywhere) responses that belong to each application	Application
Number of outbound 606 responses	<code>getTotalOutboundResponses(applicationName, "606");</code>	The number of outbound 606 (Not Acceptable Anywhere) responses that belong to each application	Application

*Table 102. SIP container task duration counters.* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration". To retrieve the attributes, use the JMXConnection.getAttribute method. For example: `_connection.getAttribute("WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration", "AvgTaskDurationOutBoundQueue")`

This table lists the task duration module counters.

Name	Attribute/Method	Description	Granularity
Average Task Duration in outbound queue	<code>AvgTaskDurationOutBoundQueue</code>	The average task duration in the SIP stack outbound queue over a configured window of time	Server
Maximum Task Duration in outbound queue	<code>MaxTaskDurationOutBoundQueue</code>	The maximum task duration in the SIP stack outbound queue over a configured window of time	Server
Minimum Task Duration in outbound queue	<code>MinTaskDurationOutBoundQueue</code>	The minimum task duration in the SIP stack outbound queue over a configured window of time	Server
Average Task Duration in processing queue	<code>AvgTaskDurationInProcessingQueue</code>	The average task duration in the SIP container processing queue over a configured window of time	Server

*Table 102. SIP container task duration counters (continued).* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration". To retrieve the attributes, use the JMXConnection.getAttribute method. For example:  
`_connection.getAttribute("WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration", "AvgTaskDurationOutBoundQueue")`

This table lists the task duration module counters.

Name	Attribute/Method	Description	Granularity
Maximum Task Duration in processing queue	MaxTaskDurationInProcessingQueue	The maximum task duration in the SIP container processing queue over a configured window of time	Server
Minimum Task Duration in processing queue	MinTaskDurationInProcessingQueue	The minimum task duration in the SIP container processing queue over a configured window of time	Server
Average Task Duration in application code	getAvgTaskDurationInApplication(String appName)	The average task duration in the SIP application code over a configured period	Application
Maximum Task Duration in application code	getMaxTaskDurationInApplication(String appName)	The maximum task duration in the SIP application code over a configured period	Application
Minimum Task Duration in application code	getMinTaskDurationInApplication(String appName)	The minimum task duration in the SIP application code over a configured period	Application

*Table 103. SIP container queue monitoring counters.* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor". To retrieve the attributes, use the JMXConnection.getAttribute method. For example:  
`_connection.getAttribute("WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor", "TotalTasksCountInProcessingQueue")`.

This table lists the queue monitoring counters.

Name	Attribute	Description	Granularity
Total number of tasks that have flowed through the processing SIP container queue	TotalTasksCountInProcessingQueue	The total number of tasks, such as messages or SIP timer events, that have flowed through the processing SIP container queue over a configured window of time	Server
Maximum number of tasks in the processing SIP container queue	PeakTasksCountInProcessingQueue	The maximum number of tasks in the processing SIP container queue over a configured window of time	Server
Minimum number of tasks in the processing SIP container queue	MinTasksCountInProcessingQueue	The minimum number of tasks in the processing SIP container queue over a configured window of time	Server



*Table 103. SIP container queue monitoring counters (continued).* The object name of the MBean from which the counters can be retrieved is: "WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor". To retrieve the attributes, use the JMXConnection.getAttribute method. For example: `_connection.getAttribute("WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor", "TotalTasksCountInProcessingQueue")`.

This table lists the queue monitoring counters.

Name	Attribute	Description	Granularity
Maximum percent full of the processing SIP container queue	PercentageFullTasksCountInProcessingQueue	The maximum percent full of the processing SIP container queue usage percentage over a configured window of time	Server
Total number of tasks that have flowed through the outbound SIP stack queue	TotalTasksCountInOutboundQueue	The total number of tasks that have flowed through the outbound SIP stack queue over a configured window of time	Server
Maximum number of tasks in the outbound SIP stack queue	PeakTasksCountInOutboundQueue	The maximum number of tasks in the outbound SIP stack queue over a configured window of time	Server
Minimum number of tasks in the outbound SIP stack queue	MinTasksCountInOutboundQueue	The minimum number of tasks in the outbound SIP stack queue over a configured window of time	Server
Maximum percent full of the outbound SIP stack queue	PercentageFullTasksCountInOutboundQueue	The maximum percent full of the outbound SIP stack queue usage percentage over a configured window of time	Server

---

## Sessions monitoring

You can use the SessionStats MBean to monitor the performance data of sessions in Liberty.

The performance data of sessions for each application is available as an MBean, which can be accessed through JMX.

The sessions that are associated with a single web application have their own SessionStats MBean (that is, one SessionStats MBean for each web application).

The ObjectName for identifying each Session MBean is:

`WebSphere:type=SessionStats,name=*`

For example:

`WebSphere:type=SessionStats,name=default_host/trade_lite`

`WebSphere:type=SessionStats,name=default_host/moneybank`

The MBean is responsible for reporting SessionStats for a single web application. The following key data is available for the SessionStats MBean after monitoring is enabled:

### CreateCount

The total number of sessions created.

### LiveCount

The total number of sessions that are currently cached in memory.

### ActiveCount

The total number of concurrently active sessions. A session is active if Liberty is processing a request that uses that session.

### InvalidatedCount

The total number of sessions that are invalidated.

### InvalidatedCountbyTimeout

The total number of sessions invalidated by a timeout.

---

## ConnectionPool monitoring

You can use the ConnectionPool MBean for ConnectionPool monitoring of Liberty.

Performance data is made available for each ConnectionPool. Connection pools manage connections from data sources and connection factories.

Each connection manager has a ConnectionPool MBean associated with it, and there is one MBean for every connection manager.

The ObjectName for identifying each ConnectionPool MBean is:

```
WebSphere:type=ConnectionPoolStats,name=<IDENTIFIER_OF_CONNECTION_MANAGER>
```

The following example shows a connection pool (for a data source or connection factory) that does not have a JNDI name. The data source [default-x] name is considered as the data source object when JNDI is not specified.

```
WebSphere:type=ConnectionPoolStats,name=transaction/dataSource[default-0]/connectionManager
```

```
<transaction enableLoggingForHeuristicReporting="true" transactionLogSize="2048">
 <dataSource transactional="false">
 <jdbcDriver libraryRef="DerbyLib"/>
 <properties.derby.embedded databaseName="<DIR Path>/<DatabaseName>" createDatabase="create"/>
 </dataSource>
</transaction>
```

Example configurations when connection manager is provided

- When an explicit ID is not specified, an ID is generated based on its parent

```
WebSphere:type=ConnectionPoolStats,name=dataSource[MyDataSource]/connectionManager[default-0]
```

```
<dataSource id="MyDataSource">
 <connectionManager maxPoolSize="10"/>
 <jdbcDriver libraryRef="DB2JCC4LIB"/>
 <properties.db2.jcc .../>
</dataSource>
```

- When an ID is specified, that becomes the identifier

```
WebSphere:type=ConnectionPoolStats,name=connectionManager[Pool2]
```

```
<dataSource id="DataSource2" jdbcDriverRef="DB2JCCDriver" connectionManagerRef="Pool2">
 <properties.db2.jcc .../>
</dataSource>
<connectionManager id="Pool2" maxPoolSize="20"/>
```

- Obtaining the correct identifier for a type 2 driver connection pool.
  - Ensure that the application that is using the pool makes a call to DB2 so that the pool is initialized.
  - Navigate to the REST interface to determine the proper identifier to use in the configuration. For example:

```
host:443/IBMJMXConnectorREST/mbeans
```

Specifying the correct identifier for a connection pool for a type 2 driver

```
{ "objectName": "WebSphere:type=ConnectionPoolStats,name=jdbc/acp01", "
 className": "com.ibm.ws.connectionpool.monitor.ConnectionPoolStats", "
 URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3Djdbc%2Facp01%
 2Ctype%3DConnectionPoolStats" }
```

The ConnectionPool MBean is responsible for reporting ConnectionPool Stats for a single connection manager. The following counter attributes are available for the ConnectionPool MBean after monitoring is enabled:

#### CreateCount

The total number of connections created.

#### DestroyCount

The total number of connections destroyed.

#### ManagedConnectionCount

The number of ManagedConnection objects that are in use.

#### WaitTime

The average waiting time in milliseconds until a connection is granted.

#### ConnectionHandleCount

The number of Connection objects that are in use.

#### FreeConnectionCount

The number of free connections in the pool.

---

## Multiple components monitoring

You can filter the components that you want to monitor by using the `monitor-1.0` feature in Liberty. The components to be filtered must be configured in the `server.xml` file.

1. To specify the components that you want to filter, add the following code to the `server.xml` file.

```
<server description="new server">
 <featureManager>
 <feature>jsp-2.2</feature>
 <feature>jdbc-4.0</feature>
 <feature>monitor-1.0</feature>
 </featureManager>
 <monitor filter="JVM,ThreadPool,WebContainer,Session,ConnectionPool"/>
</server>
```

By default, if the filters are not provided in the `<monitor>` tag, all the components that are currently monitored as part of the `monitor-1.0` feature are monitored. You can specify the components that you want to monitor by providing the group name as part of the filter.

For example: If you want to monitor only the *JVM* and *WebContainer* components, specify the components in the `server.xml` file as follows:

```
<monitor filter="JVM,WebContainer"/>
```

2. To remove components from monitoring.

To stop monitoring a component, you must remove the component from the filter group at run time.

For example: The following filter configuration monitors the *JVM*, *ThreadPool*, *WebContainer*, *Session* and *ConnectionPool* components:

```
<monitor filter="JVM,ThreadPool,WebContainer,Session,ConnectionPool"/>
```

To stop monitoring the components *WebContainer* and *Session*, remove those components from the filter configuration:

```
<monitor filter="JVM,ThreadPool,ConnectionPool" />
```

3. To enable monitoring of components at run time.

If you want to enable monitoring for specific components at run time, you can specify the components in the monitor tag at run time.

The data that is collected by the filtering components is available as MXBeans. For more information about the various MXBeans, see Chapter 10, “Monitoring the Liberty server runtime environment,” on page 1423.

**Note:** Currently, fine-grained monitoring is supported only at the component level (such as *WebContainer*, *ThreadPool*, *JVM*) and not at the counter level.

---

## HTTP access logging

You can configure access log settings for HTTP endpoints.

### HTTP access log settings

An HTTP access log contains a record of all inbound client requests handled by HTTP endpoints. You can enable access logging in the HTTP server, or you can enable it in the Liberty server in two modes: one log common to multiple endpoints, or one log for each endpoint.

**Note:** If you do not specify attributes, the defaults are used. To see a list of the default attributes, see `httpAccessLogging` in the list of liberty configuration elements: \*\*\*\* MISSING FILE \*\*\*\*.

- Using a common log

To enable logging for multiple endpoints using common settings, include *httpAccessLogging* as a top-level element in your `server.xml` file, and then reference it from multiple `httpEndpoint` elements:

```
<httpAccessLogging id="accessLogging"/>
<httpEndpoint id="defaultHttpEndpoint" accessLoggingRef="accessLogging"/>
<httpEndpoint id="otherHttpEndpoint" accessLoggingRef="accessLogging" httpPort="9081" httpsPort="9444"/>
```

- Using distinct logs for each endpoint

To enable logging for individual endpoints, use an *accessLogging* child element and specify a file path that does not conflict with other logs:

```
<httpEndpoint id="defaultHttpEndpoint">
 <accessLogging filepath="${server.output.dir}/logs/http_defaultEndpoint_access.log"/>
</httpEndpoint>
```

- Using logs for the HTTP server

For a list of the available HTTP server side properties and their descriptions, see `Apache Module mod_log_config`.

### HTTP access log format

For a list of the available log format properties and their descriptions, see the *accessLogFormat* configuration for WebSphere Application Server traditional in HTTP transport channel custom properties. This log format string is specified using the *logFormat* attribute of `httpAccessLogging` or `accessLogging` elements in `server.xml`:

```
<httpAccessLogging logFormat='%h %u %{t}W "%r" %s %b' />
```

or

```
<httpEndpoint id="defaultHttpEndpoint">
 <accessLogging filepath="${server.output.dir}/logs/http_defaultEndpoint_access.log"
 logFormat='%h %i %u %t "%r" %s %b' />
</httpEndpoint>
```

---

## Chapter 11. Tuning Liberty

You can tune parameters and attributes of Liberty.

### About this task

Liberty supports different attributes in the `server.xml` file to influence application performance. You can use these parameters and attributes to achieve better performance. To tune Liberty for secure applications, see [Tuning Liberty for secure applications](#).

### Procedure

- Tune the JVM.

Tuning the JVM is a most important tuning step whether you configure a development or production environment. When you tune the JVM for Liberty, use the `jvm.options` file in the `${server.config.dir}` directory. You can specify each of the JVM arguments that you want to use, one option per line. For more information, see “Customizing the Liberty environment” on page 947. An example of the `jvm.options` file is as follows:

```
-Xms50m
-Xmx256m
```

For a development environment, you might be interested in faster server startup, so consider setting the minimum heap size to a small value, and the maximum heap size to whatever value is needed for your application. For a production environment, setting the minimum heap size and maximum heap size to the same value can provide the best performance by avoiding heap expansion and contraction.

- Tune transport channel services.

The transport channel services manage client connections, I/O processing for HTTP, thread pools, and connection pools. For applications on Liberty, the following attributes are available for different elements that can be used to improve runtime performance, scalability, or both. For each of these attributes, see `**** MISSING FILE ****`.

#### **maxKeepAliveRequests of httpOptions**

This option specifies the maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of `-1` means unlimited. This option supports low latency or high throughput applications, and SSL connections for use in situations where building up a new connection can be costly. Here is an example of how you code this option in the `server.xml` file:

```
<httpOptions maxKeepAliveRequests="-1" />
```

#### **maxPoolSize of connectionManager**

This option specifies the maximum number of physical connections for the connection pool. The default value is 50. The optimal setting here depends on the application characteristics. For an application in which every thread obtains a connection to the database, you might start with a 1:1 mapping to the `coreThreads` attribute. Here is an example of how you code this option in the `server.xml` file:

```
<connectionManager ... maxPoolSize="40" />
```

#### **purgePolicy of connectionManager**

This option specifies which connections to destroy when a stale connection is detected in a pool. The default value is the entire pool. It might be better to purge only the failing connection. Here is an example of how you code this option in the `server.xml` file:

```
<connectionManager ... purgePolicy="FailingConnectionOnly" />
```

#### **numConnectionsPerThreadLocal of connectionManager**

This option specifies the number of database connections to cache for each executor thread.

This setting can provide a major improvement on large multi-core (8+) machines by reserving the specified number of database connections for each thread.

Using thread local storage for connections can increase performance for applications on multi-threaded systems. When you set **numConnectionsPerThreadLocal** to 1 or more, these connections per thread are stored in thread local storage. When you use **numConnectionsPerThreadLocal**, consider two other values:

- The number of application threads
- The connection pool maximum connections

For best performance, if you have n applications threads, you must set the maximum pool connections to at least n times the value of the **numConnectionsPerThreadLocal** attribute and you must use the same credentials for all connection requests. For example, if you use 20 application threads, set the maximum pool connections to 20 or more; If you set the value of **numConnectionsPerThreadLocal** attribute as 2 and there are 20 application threads, set the maximum pool connection to 40 or more. Here is an example of how you code this option in the `server.xml` file:

```
<connectionManager ... numConnectionsPerThreadLocal="1" />
```

#### **statementCacheSize of dataSource**

This option specifies the maximum number of cached prepared statements per connection. To set this option, complete the following prerequisite:

- Review the application code (or an SQL trace that you gather from the database or database driver) for all unique prepared statements.
- Ensure that the cache size is larger than the number of statements.

Here is an example of how you code this option in the `server.xml` file:

```
<dataSource ... statementCacheSize="60" >
```

#### **isolationLevel of dataSource**

The data source isolation level specifies the degree of data integrity and concurrency, which in turns controls the level of database locking. Four different options are available as following in order of best performing (least integrity) to worst performing (best integrity).

##### **TRANSACTION\_READ\_UNCOMMITTED**

Dirty reads, non-repeatable reads, and phantom reads can occur.

##### **TRANSACTION\_READ\_COMMITTED**

Dirty reads are prevented; non-repeatable reads and phantom reads can occur.

##### **TRANSACTION\_REPEATABLE\_READ**

Dirty reads and non-repeatable reads are prevented; phantom reads can occur.

##### **TRANSACTION\_SERIALIZABLE**

Dirty reads, non-repeatable reads, and phantom reads are prevented.

Here is an example of how you code this option in the `server.xml` file:

```
<dataSource ... isolationLevel="TRANSACTION_READ_COMMITTED">
```

- **8.5.5.6** Tune the default executor.

The Liberty default executor is self-tuning and adapts to the current workload by dynamically adding or removing threads. For most workloads, the executor does not require any tuning, and you are advised not to change any settings of the executor unless you encounter specific problems with thread creation.

If necessary, you can configure the **coreThreads** and **maxThreads** parameters of the executor element in the `server.xml` file to set lower and upper bounds for the Liberty auto-tuning code. The **coreThreads** setting is not usually needed because the executor contains aggressive anti-deadlocking code that adds threads to break the executor out of deadlock scenarios. Rarely, the anti-deadlocking code adds more

threads than are required. In this situation, you can use the **maxThreads** parameter of the executor element to cap the number of threads the executor is allowed to create.

- Decrease response time of servlets.

To decrease response time of servlets, add the following attribute to the `server.xml` file:

```
<webContainer skipMetaInfResourcesProcessing="true"/>
```

- Reduce idle server CPU time.

To reduce idle server CPU time, add the following attributes to the `server.xml` file:

```
<applicationMonitor dropinsEnabled="false" updateTrigger="disabled"/>
<config updateTrigger="disabled"/>
```

When the attributes are added, your server no longer monitors for configuration or application updates.

For more information about the configuration element descriptions, see **\*\*\*\* MISSING FILE \*\*\*\***.

- **8.5.5.7** Tuning startup time.

### CDI 1.2

By default, the CDI 1.2 feature scans all application archives. The CDI 1.2 feature can increase startup time substantially and have the most effect on larger applications. Implicit archive scanning for annotations can be disabled by setting **enableImplicitBeanArchives** value to `false`. This setting skips the scanning of archives unless they contain a `beans.xml` file.

```
<cdi12 enableImplicitBeanArchives="false"/>
```

**Note:** The `cdi-1.2` feature may be included even if its not in the `<features>` section of your `server.xml` file because other features, such as `webProfile-7.0` and `javaee-7.0`, include the `cdi-1.2` feature. Look in the `messages.log` file for "The server installed the following features:" to see if `cdi-1.2` was installed.

---

## Tuning Liberty for secure applications

You can tune Liberty to maximize the performance for secure applications.

### About this task

When you are securing your WebSphere application environment, it is important to understand the impact security can have on performance. Within an application server environment, running applications with security settings can often decrease performance because of increased processor usage from security tasks such as encryption, authentication, and authorization. These services can often increase the path length of application server requests, requiring more resources for each request and slowing down application throughput.

In most cases, you can reduce or eliminate some of this security-related performance loss through performance tuning. You can adjust the resources that are used by security services and choose to use only the security services that are required by a particular application or environment. Achieving the best possible performance, without sacrificing any necessary security, requires an understanding of your network topology and the security needs of your applications.

### Procedure

- Choose the connections that you want to encrypt.

In a WebSphere Application Server environment, you can encrypt the following transports:

- HTTP traffic that is to the web server
- HTTP traffic that is from the web server to the application server
- SOAP/JMX traffic
- File transfer service

- Web services over HTTP

When you are determining which traffic to transport over encrypted connections, consider whether the network that is connecting the communicating machines is private or public. There is a significant amount of resources that are associated with setting up a secured connection and encrypting and decrypting the traffic over that connection. You can make significant performance improvements by not requiring encryption over a secured network, for instance. If your application does not require that traffic is encrypted from the client to the HTTP server and from the HTTP server to the application Server, you might be able to use SSL only from the client to the HTTP server and therefore reduce resources that are required by security.

- Enable on-chip Advanced Encryption Standard (AES) Encryption.

If you are using IBM® SDK Java Technology Edition Version 7, Service Refresh 3 or later, and you are running on an Intel processor that supports the Advanced Encryption Standard (AES) New Instructions (AES-NI) instruction set, you can achieve performance improvement by taking advantage of on-chip AES encryption. Using these features, you can run AES encryption and decryption using hardware instructions without extra software.

To enable AES-NI usage, add the following property to the JVM command line or `jvm.options` file:

```
com.ibm.crypto.provider.doAESInHardware=true
```

Add the following property to the JVM command line or the `jvm.options` file to verify that the processor supports AES-NI instruction set:

```
com.ibm.crypto.provider.AESNITrace=true
```

For more information, see Intel Advanced Encryption Standard New Instructions.

- Choose your cipher key length.

In some cases, the bit length of a cipher key is governed by regulations that are put on the transfer of certain types of data. In these cases, the cipher and key length that you choose for a particular SSL connection can be predetermined. For situations where the key length is not regulated, you must choose the appropriate resources to allocate to security so that performance is not decreased more than is necessary. For example, a 256-bit cipher offers stronger encryption than a corresponding 128-bit cipher. However, decrypting messages with a stronger cipher requires more processing time.

When you are making your decision about the encryption strength to choose, consider the type of data that is traveling across a network. For example, sensitive information, such as financial or medical records, needs the maximum amount of security. Also, consider who has access to the network. If the network is protected by a firewall, consider decreasing the strength of the cipher, or possibly transmitting the data over an decrypted connection.

For more information about configuring SSL settings on Liberty, see “SSL configuration attributes” on page 1153

- Set connection keep-alive requests.

In the Secure Socket Layer (SSL) protocol, the initial handshake uses a public-key cipher to exchange a private key for a faster private-key cipher. This faster cipher is used to encrypt and decrypt communication beyond the initial handshake. Because the cipher used for subsequent communication is faster than the one used for the initial handshake, it is important from a performance perspective to limit the number of SSL handshakes that are performed within the application server. You can achieve this result by increasing the length of an SSL connection by increasing session affinity.

One way to increase the duration of a single SSL connection is to enable persistent HTTP keep-alive connections. Increasing the duration prevents SSL handshakes from taking place on consecutive requests. You can make sure that persistent connections are enabled by verifying that the `keepAliveEnabled` attribute in the `httpOptions` element is set to `true` in the `server.xml` file. The default value is `true`.

Another way to tune persistent connections is by setting the maximum number of consecutive requests on a single HTTP connection. If your clients are making more than 100 requests consecutively, consider increasing the value of the `maxKeepAliveRequests` attribute in the `httpOptions` element in the `server.xml` file. The default value is 100.

For more information, see \*\*\*\* MISSING FILE \*\*\*\*.



- Set the authentication cache settings.

Because the creation of an authentication subject can increase processor usage, Liberty provides an authentication cache to store a subject after the authentication of a user is successful. To fully take advantage of this service to increase performance, you must make sure that it is turned on and tuned according to your users and applications.

- Make sure that you do not disable the authentication cache. By default, the authentication cache is enabled to help improve performance.
- Consider changing the authentication cache timeout value. Increasing the timeout value enables subjects to remain in the authentication cache longer and reduces the number of reauthentications needed. However, increasing the timeout value increases the risk of user permissions becoming stale compared to a modified external repository, such as LDAP. Set your authentication cache timeout to reflect the estimated length of client sessions. You can specify the cache timeout by setting the value of the timeout attribute to whichever time you choose in the authCache element in the server.xml file. The default value is 600 seconds.
- Finally, if you are experiencing authentication times longer than expected, or you are noticing more traffic to an external authentication repository than expected, the authentication cache might be full. When the authentication cache is full, subjects are evicted. There is not a one-to-one mapping of authenticated users to authentication cache entries. The number of entries in the cache per user depends on other security configurations. It is a best practice for the maximum size of the authentication cache to be larger than the number of distinct authenticated users that are accessing the server at one time. Setting the maximum size of the authentication cache this way helps prevent subjects from being evicted from the cache before timing out. You can change the maximum size of the authentication cache by setting the value of the maxSize attribute in the authCache element in the server.xml file. The default size is 25000.

For more information, see “Configuring the authentication cache in Liberty” on page 1183.

- Configure the HTTP session affinity settings.

One of the operations in a secure application environment that drains performance the most is the initial setup, including the SSL handshake and authentication. In clustered environments, performance decreases might occur when a web client accesses different application servers. To prevent the increased processor usage of the SSL handshake and reauthentication, it is important to make sure that HTTP session affinity is configured.

HTTP session affinity ensures that consecutive client requests are routed to the same application server. HTTP session affinity aids performance in various ways, but specifically, it prevents the increased processor usage of reauthentication and SSL handshaking. Consult the documentation of your HTTP Server or Load Balancer for instructions on setting up HTTP session affinity.

For more information, see “Configuring session persistence for Liberty” on page 990.

---

## Tuning federated LDAP repositories in Liberty

### 8.5.5.1

You can improve the performance of the federated LDAP repositories by monitoring and adjusting the cache and the context pool elements in the server.xml file.

#### About this task

The cached query results of the LDAP repositories save time, because the data need not be retrieved from the back-end server every time an LDAP operation is performed. The LDAP cache attributes are stored in the <ldapCache> element for quicker access. You must monitor the status of the cache and adjust the cache control parameters to improve the performance of the cache. The context pooling parameters can be adjusted to improve the performance of concurrent accesses to the LDAP servers.

## Procedure

- Configure the <ldapCache> element in the server.xml file.

Specify the LDAP cache control parameters to improve the performance:

### attributesCache

<size>: Specifies the number of entities that are stored in the cache. You can increase the size of the cache based on your business requirement, for example, increase the cache size if more number of entities are required in a business scenario.

<timeout>: Specifies how long the results can be cached before they are invalidated. If the back-end LDAP data is refreshed frequently to maintain an up-to-date cache, set a lesser timeout duration value.

<sizeLimit>: Specifies the maximum number of LDAP attributes per entity that can be stored in the cache. If an entity is associated with many attributes, increase the <sizeLimit> value.

### searchResultSizeLimit

Specifies the maximum number of search results that can be stored in the cache. Use the parameters in the <searchResultSizeLimit> element to tune the search results that are returned as part of the query.

- Configure the <contextPool> element parameters in the server.xml file to improve the performance of concurrent access to an LDAP server.

You can adjust the following parameters in the <contextPool> element to control the cache:

### contextPool

<initialSize>: Specifies the initial size of the context pool. The value must be set based on the load on the repository. If the initial number of requests to the LDAP server is expected to be high, increase the value of the initial size.

<maxSize>: Specifies the maximum context pool size. The value must be set based on the load on the repository. If you want to restrict the number of connections to the LDAP server, then set the value of the <maxSize> element to less than half of the maximum number of connections that the LDAP server can handle.

<timeout>: Specifies the duration after which the context pool times out. Specify a shorter timeout value so that fresh connections can be made to the LDAP server after the specified duration is timed out. For example, if the established connection is timed out after the configured interval, then set a shorter duration than the firewall timeout duration so that the connection is re-established.

<waitTime>: Specifies the waiting time before the context pool times out. If the value specified is high, then the time that is taken to establish a connection to the LDAP server is increased accordingly.

For more information about the <ldapCache> and <contextPool> elements, see \*\*\*\* MISSING FILE \*\*\*\*

---

## Chapter 12. Troubleshooting tips

Tips for troubleshooting Liberty.

To help you identify and resolve problems, the product has a unified logging component. See “Logging and Trace” on page 1461. You can also use the IBM Support Assistant Data Collector (ISADC) command tool in the `${wlp.install.dir}/bin` directory to quickly collect diagnostic files, such as log files, configuration files or to run traces.

If you receive an exception message, information about the message is available in “Messages” on page 1490.

The Java API documentation for each Liberty API is detailed in the Programming Interfaces (APIs) section of the documentation, and is also available as a separate .zip file in one of the javadoc subdirectories of the `${wlp.install.dir}/dev` directory.

**Distributed operating systems** Details of the main known restrictions that apply when you use Liberty are provided in the following two topics:

- “Runtime environment known issues and restrictions” on page 1480.
- **Distributed operating systems** “Developer Tools known issues and restrictions” on page 1488.

**IBM i** For details of the main known restrictions that apply when using Liberty, see “Runtime environment known issues and restrictions” on page 1480.

Here is a set of tips to help you troubleshoot commonly experienced problems:

- Troubleshooting JDKs
- “Troubleshooting security” on page 1452
- “Troubleshooting LDAP” on page 1454
- “Troubleshooting SSL” on page 1454
- “Troubleshooting CORBA/IIOP” on page 1455
- “Troubleshooting logging and tracing” on page 1456
- **Distributed operating systems** **IBM i** “Applying fix packs and interim fixes to an archive install” on page 1456
- “Troubleshooting performance” on page 1456
- **8.5.5.7** Troubleshooting SAML

### Check that your JDKs are at a supported level

If you experience problems that are not readily explained, check that the JDKs you are using are compliant with Java Version 6 or later, and are at a current service level. See “Minimum supported Java levels” on page 1481.

**Note:** A deadlock can occur when using Oracle based JVMs using Java Version 6. If you are using an affected JVM or JDK, the following settings can help prevent the deadlock from occurring:

- Enable the following VM option: `-XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass`
- Set the Equinox framework option to use classname locking for classloading by setting the following Equinox configuration option: `-Dosgi.classloader.lock=classname`

These can be set in a Java properties file, for example `jvm.options`, when starting the Liberty server.

## Troubleshooting security

This section describes some common security problems and solutions you can choose.

### **SESN0008E: A user authenticated as anonymous has attempted to access a session owned by user:LdapRegistry/cn=steven,o=myCompany,c=US.**

This error happens when an unauthenticated user tries to access a session created by an authenticated user. Session security that is enabled by default prevents unauthorized access of the sessions. Only the user who created a session can access it. See session security for more information.

This error can happen when you use a JSP (login.jsp for example) for your form-login and the SSO token sent by the browser is expired. Because the SSO token is expired, the user is prompted to log in again using the login.jsp page configured for the form-login. The jsp page, by default, tries to get a session. This session was originally created by the user whose token is expired. However, the token is expired and the user is not authenticated, no credentials are established when accessing this session that results in this error.

To avoid this error, restart the browser that starts a new session, or configure the login.jsp file to not create the session by default. If you choose to update the login.jsp file, set `<%@ page session="false" %>`.

### **CWWKS9104A: Authorization failed for user {0} while invoking {1} on {2}. The user is not granted access to any of the required roles: {3}.**

This error occurs when you do not have authorization to the roles required by the application. Make sure that the user or the group it belongs to is mapped to at least one of the roles that are mentioned in the error message. A user-to-role mapping defined in the `ibm-application-bnd.xml` file takes precedence over a mapping defined in the `server.xml` file. Check both resources to ensure that the correct mapping is defined. See “Configuring authorization for applications in Liberty” on page 1254.

### **CWWKS9104A: Authorization failed for user {0}.**

This error can occur if you specify both an application and webApplication for the same context root. If a conflict happens the latest configuration that is defined is ignored and causes an unexpected error, such as CWWKS9104A.

### **CWWKZ0013E: It is not possible to start two applications called {0} followed by unexpected security behavior and error messages such as CWWKS9104A.**

This error occurs when you specify your application in both the `server.xml` by using the application element and in the `dropins` folder. As a result, the application is attempted to be installed twice and the security configuration in the `server.xml` file might or might not take effect. To fix this, you must remove your application from the `dropins` folder and copy it to another directory. If you have to leave it in the `dropins` folder, you must disable the application monitoring by using the following code in your `server.xml` file:

```
<applicationMonitor dropinsEnabled="false"/>
```

### **An unauthenticated user was able to access my servlet or JSP.**

A user with a principal of **UNAUTHENTICATED** (or the unauthenticated SAF user on z/OS) is created when authentication fails or when your servlet or JSP is unprotected. An unauthenticated user can access your servlet or JSP if you do not define any security constraints or if you map the **EVERYONE** special subject to the role required by your application. Review the user-to-role mappings in the `ibm-application-bnd.xml/xml` and `server.xml` files. Take one of the following options:

- If your servlet or JSP is unprotected, add security constraints to the deployment descriptor of your application. See “Authentication” on page 585.
- If you do not want any unauthenticated user to access your application, remove the **EVERYONE** special subject from the mapping for that role. See “Configuring authorization for applications in Liberty” on page 1254.

- If a certain user cannot be authorized to your servlet or JSP, review who is mapped to that role by examining the `ibm-application-bnd.xml/xml` and `server.xml` files. You can map a role to a user, group, or special subject. If your role is mapped to the **EVERYONE** special subject, any user is granted access. If your role is mapped to the **ALL\_AUTHENTICATED** special subject, any authenticated user is granted access. Remove these special subjects if you want to further limit who can access your servlet or JSP. Finally, check what group the user belongs to. Although the user might not have explicit access, the group might be mapped to the role. In this case, remove the user from the authorized group or remove the group from role mapping. See “Configuring authorization for applications in Liberty” on page 1254.

### Single sign-on (SSO) does not work.

If SSO does not work, make sure that the different Liberty servers that use the same LTPA keys, password, and `ssoCookieName` attribute of `webAppSecurity` element have the same Universal Time (UTC) and share the same user registry. Also, if the token expires or if the cookie is sent from a web browser after changing the user registry in a manner that is inconsistent, such as modifying the realm or removing the user the cookie represents, the SSO fails and the user is prompted to enter the credential information again. See “Customizing SSO configuration using LTPA cookies in Liberty” on page 1203.

### Debugging security public APIs.

`WSSecurityHelper` and `RegistryHelper` are loaded even if security is not enabled, for example, if a security feature - `appSecurity-1.0,appSecurity-2.0` or `zosSecurity-1.0` - is not specified. If security is not enabled, then `WSSecurityHelper.isServerSecurityEnabled()` and `WSSecurityHelper.isGlobalSecurityEnabled()` methods both return false, and `RegistryHelper.getUserRegistry()` method returns null.

Other security public API classes might not be loaded if security is not enabled. If you try to access these classes and call a method on one of these classes, you might get a `java.lang.NoClassDefFoundError` exception.

To avoid getting `java.lang.NoClassDefFoundError` exceptions, you must first test to see whether security is enabled by calling the `WSSecurityHelper.isServerSecurityEnabled()` or `WSSecurityHelper.isGlobalSecurityEnabled()` class, and then call other security public API classes only when security is enabled. See “Security public APIs” on page 611 for examples of this coding technique.

**Note:** This behavior is different from the traditional. In traditional, all classes are always loaded regardless of whether security is enabled or not.

### Cannot authenticate users with Unicode characters

In order to authenticate users whose names contain Unicode characters, you must set the character encoding type used by the Liberty server to UTF-8 by adding the following JVM option to the server start command:

```
-Dclient.encoding.override=UTF-8
```

You must also specify the charset and pageEncoding in your login page. Here is an example for specifying these parameters on a login JSP page:

```
<%@page contentType="text/html"; charset=UTF-8" pageEncoding="UTF-8"%>
```



### 8.5.5.4 `java.lang.annotation.AnnotationFormatError: java.lang.IllegalArgumentException:Wrong type at constant pool index at sun.reflect.annotation.AnnotationParser.parseAnnotations(AnnotationParser.java:87)`

This error can occur when an OpenID Connect or OAuth provider uses a Database Store for client registration with some JDK 7 versions.

To avoid this problem, upgrade to JDK version 7.1.

## Troubleshooting LDAP

This section describes some common LDAP problems and solutions you can choose.

**FFDC1015I: An FFDC Incident has been created: javax.naming.ServiceUnavailableException: myldapserver.mycompany.com:636; socket closed  
com.ibm.ws.security.registry.ldap.internal.LdapRegistry 298**

This message in messages.log indicates that the configured LDAP server cannot be reached. Check your LDAP server to verify that it is running and that its IP address can be accessed from Liberty server.

**The javax.naming.CommunicationException: simple bind failed: myldapserver.mycompany.com:636 [Root exception is javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.g: PKIX path building failed: java.security.cert.CertPathBuilderException: unable to find valid certification path to requested target]**

If you enable SSL on your LDAP server without copying the signer of the LDAP server into the truststore referenced in the LDAPSSLSettings element, a connection with the LDAP server fails with an SSL handshake error. Make sure that you copy the signer of the LDAP server into your truststore.

**The javax.naming.CommunicationException: myldapserver.mycompany.com:389 [Root exception is java.net.BindException: Address already in use: connect]**

This message might appear in the FFDC logs and indicates that the usable sockets on the local server are exhausted, which can result in a failure when connecting to your LDAP server. Make sure that the socket is not used and try again.

**CWWKS1100A: Authentication did not succeed for user ID xxxxx. An invalid user ID or password was specified**

This FFDC exception might happen on the server even though the user mentioned in the message is a valid user on the LDAP server under heavy load. With the LDAP configuration, you can add the reuseConnection=false property or disable it by using the developer tools. To fix the problem, set this property to the default value of true.

## Troubleshooting SSL

This section describes some common SSL problems and solutions you can choose.

**CWWKS9105E: Could not determine the SSL port for automatic redirection.**

This error occurs when you try to access an application that redirects to an SSL port and the SSL port is not available. The port might not be available because of a missing SSL configuration or some error in the SSL configuration definition. Check the SSL configuration in the server.xml file to make sure that it exists and is correct. See “Securing communications in Liberty” on page 1151.

**FFDC1015I: An FFDC Incident has been created: “java.lang.IllegalArgumentException: Unknown, incomplete configuration: missing id field  
com.ibm.ws.config.internal.cm.ManagedServiceFactoryTracker aSyncReadNupdate. Exception thrown while trying to read configuration and update ManagedServiceFactory. Exception = java.lang.IllegalArgumentException: Unknown, incomplete configuration: missing id field” at  
ffdc\_12.04.18\_16.09.42.0.log**

This error occurs when a keystore element exists in the configuration without an ID field. If you use a minimal SSL configuration, set the ID field to defaultKeyStore.

**You might get an exception if using a LDAP user registry with sslEnabled and a sslRef value is not specified.**

For example, a configuration has sslEnabled set to true but there is not a sslRef value, as shown in the following example:

```
<ldapRegistry id="ldap" realm="SampleLdapIDSRealm"
host="ccwin12.austin.ibm.com" port="636" ignoreCase="true"
baseDN="o=ibm,c=us"
bindDN="cn=root"
```

```
bindPassword="rootpwd"
ldapType="IBM Tivoli Directory Server"
idsFilters="ibm_dir_server"
sslEnabled="true"
searchTimeout="8m" />
```

You must enter a `sslRef` value. If you are using a minimal SSL configuration that is similar to the following:

```
<ltkeyStore id="defaultKeyStore" location="key.jks"
password="mypassword" />
```

the `sslRef` field should be set to `defaultSSLConfig`.

If a custom SSL configuration is configured, the name of that configuration should be placed in the `sslRef` field.

### If you use a JDK from the WebSphere Application Server, you might see the following error if SSL is enabled on your Liberty Server.

```
java.net.SocketException: java.lang.ClassNotFoundException: Cannot find the specified class com.ibm.websphere.ssl.protocol.SSLSocketFactory
 at javax.net.ssl.DefaultSSLSocketFactory.a(SSLSocketFactory.java:11)
 at javax.net.ssl.DefaultSSLSocketFactory.createSocket(SSLSocketFactory.java:6)
 at com.ibm.net.ssl.www2.protocol.https.c.afterConnect(c.java:161)
 at com.ibm.net.ssl.www2.protocol.https.d.connect(d.java:36)
 at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1184)
 at java.net.HttpURLConnection.getResponseCode(HttpURLConnection.java:390)
 at com.ibm.net.ssl.www2.protocol.https.b.getResponseCode(b.java:75)
 at com.ibm.ws.jmx.connector.client.rest.internal.RESTMBeanServerConnection.loadJMXServerInfo(RESTMBeanServerConnection.java:114)
 at com.ibm.ws.jmx.connector.client.rest.internal.RESTMBeanServerConnection.<init>(RESTMBeanServerConnection.java:114)
 at com.ibm.ws.jmx.connector.client.rest.internal.Connector.connect(Connector.java:315)
 at com.ibm.ws.jmx.connector.client.rest.internal.Connector.connect(Connector.java:103)
```

This error occurs because the WebSphere Application Server SSL socket factories are not supported by Liberty. You can get past this problem by taking the following steps:

- Create a text file, such as `my.java.security` with the following two empty entries
 

```
ssl.SocketFactory.provider=
ssl.ServerSocketFactory.provider=
```
- Create a `jvm.options` file for your Liberty server
- Add the following property to your `jvm.options` file, that includes the full path to your text file that you just created
 

```
-Djava.security.properties=fullPathTo/my.java.security
```
- If you want to make this more reusable, you can put the `my.java.security` file in your server directory, and then you will be able to use a relative path like this:
 

```
-Djava.security.properties=./my.java.security
```

For more information, see “Customizing the Liberty environment” on page 947.

## Troubleshooting CORBA/IIOP

This section describes some common CORBA problems and solutions you can choose.

### If you use a JDK from the WebSphere Application Server, you might see the following error if your application uses CORBA/IIOP communications.

```
15:21:58.096 com.ibm.rmi.pi.InterceptorManager runPreInit:178 Init Process ORBRas [default] java.lang.ClassNotFoundException: com.ibm.ISecurityLocalObjectBaseL13Impl.CSIClientRI
 at com.ibm.CORBA.iiop.UtilDelegateImpl.loadClass(UtilDelegateImpl.java:685)
 at javax.rmi.CORBA.Util.loadClass(Util.java:246)
 at com.ibm.rmi.pi.InterceptorManager.runPreInit(InterceptorManager.java:172)
 at com.ibm.rmi.corba.ORB.initializeInterceptors(ORB.java:664)
 at com.ibm.CORBA.iiop.ORB.initializeInterceptors(ORB.java:1084)
 at com.ibm.rmi.corba.ORB.orbParameters(ORB.java:1359)
```

```
at com.ibm.rmi.corba.ORB.set_parameters(ORB.java:1271)
at com.ibm.CORBA.iiop.ORB.set_parameters(ORB.java:1694)
at org.omg.CORBA.ORB.init(ORB.java:371)
...
```

This error occurs because the WebSphere Application Server Object Request Broker (ORB) interceptors are not supported by Liberty. You can resolve this problem by editing the `orb.properties` file from the JDK to not use these interceptors. This file is usually found in the WebSphere `<JAVA_HOME>/jre/lib` directory, though you might have overridden it with a copy in the user's `<USER_HOME>` directory. The following example shows the lines in the `orb.properties` file that must be commented out:

```
WS Interceptors
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.Transaction.JTS.TxInterceptorInitializer
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ejs.ras.RasContextSupport
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.ClientRIWrapper
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.activity.remote.cos.ActivityServiceClientInterceptor
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.CSIClientRI
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.debug.olt.ivbtrjrt.OLT_RI
#org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.wlm.client.WLMClientInitializer

WS ORB & Plugins properties
#com.ibm.ws.orb.transport.ConnectionInterceptorName=com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInter
```

## Troubleshooting logging and tracing

This section describes some common problems with logging and tracing.

### The `java.util.logging` -- Java logging programming interface.

Liberty does not support using the `logging.properties` file to configure `java.util.logging`. Use Java code, for example in a deployed application or user feature, to create and add `java.util.logging` handlers, filters, or formatters.

Since the Liberty server manages the `java.util.logging` logger levels in accordance with the `traceSpecification` attribute of the logging configuration element, you should avoid using the `Logger.setLevel` method.

## Distributed operating systems

IBM i

## Applying fix packs and interim fixes to an archive install

If you installed your Liberty runtime environment from an archive file, rather than by using Installation Manager, you must take special measures when you apply service updates. For more information, see “Applying a fix pack to a Liberty Java archive installation” on page 840 and “Applying an interim fix to a Liberty archive installation” on page 849.

## Troubleshooting performance

This section describes some common performance problems and solutions you can choose.

### High CPU usage by your application monitor.

This error can occur if your application monitor has many files under the `apps/` directory and is polling too frequently.

To avoid this problem there are a number of things you can change.

1. Increase the value of the `pollingRate` attribute.
2. Update the `server.xml` to include an `applicationMonitor` element with an `updateTrigger` that is not polled.

```
server.xml
<applicationMonitor updateTrigger="mbean" />
```

3. Reduce the number of files under the `apps/` directory.



For more information about the `applicationMonitor` element, see “Controlling dynamic updates” on page 975.

8.5.5.7

## Troubleshooting SAML

This section describes some common problems with SAML and the solutions you must apply.

### **`java.lang.ArrayIndexOutOfBoundsException: Array index out of range: 0`**

This exception can occur when attempting multiple logins through an unsolicited Service Provider (SP) initiated request without removing the Identity Provider token (IdP).

To avoid this, add `<httpSession invalidateOnUnauthorizedSessionRequestException="true" />` in the relevant `unsolicited server.xml` file.

### **`java.lang.IllegalStateException: CWWKS0010E: While getting the caller principal, the caller subject was found to have more than one principal of type WSPincipal. Only one WSPincipal can exist in the subject. The names of the WSPincipals are: {}`**

This exception can occur if a SAML user has previously logged directly into an on-premises user registry. To avoid this problem, a SAML user must not directly login to an on-premises user registry.

---

## Security bulletins for the Liberty profile

To avoid preventable security issues, stay up to date on the most current maintenance options for the product.

### Subscription to security bulletins

Subscribe to My Notifications to receive notifications of security bulletins for WebSphere Application Server. These notifications include important product support alerts.

### Update Strategy

Refer to the Update Strategy for fix packs. Take special note of the recommended update path.

### Security fixes

Refer to the security fixes for WebSphere Application Server traditional, WebSphere Application Server Liberty, IBM HTTP Server, and Java.

---

## Logstash collector

8.5.5.9

Liberty generates various events at runtime, such as log events, trace events, first failure data capture (ffdc) events, access log events, and garbage collection events. It is helpful to consolidate events from all servers so the events can be searched, filtered, and analyzed, particularly when you are managing many servers, or when you are running servers on different platforms (for example on dedicated hardware and in the cloud). It can also be helpful to store events on a separate server in cases where you use Liberty in environments that lack persistent file storage for problem determination data. Liberty now provides the Logstash collector feature to help you remotely consolidate events. You can avoid running the agents on your Liberty server machine to collect your events using this feature. The collector captures in-flight events, breaks them into fields, and securely forwards the events to the configured Logstash server.

## Logstash collector

The Logstash collector feature (logstashCollector-1.0) sends events to a Logstash server that you provide.

The logstash collector feature offers a flexible way to choose one or more of the following supported sources of data that needs to be sent to logstash:

- message - messages log events
- trace - traces log events
- accesslog - Http access log events
- ffdc - FFDC log events
- garbageCollection - GarbageCollection events

Logstash can be used with the Elasticsearch search server and Kibana dashboard, all of which you provide, set up, and manage, to give a consolidated view of logs or other events from across your enterprise. Note that there are no separate processes or agents to be set up on the Liberty server machine when using the Logstash collector to forward events.

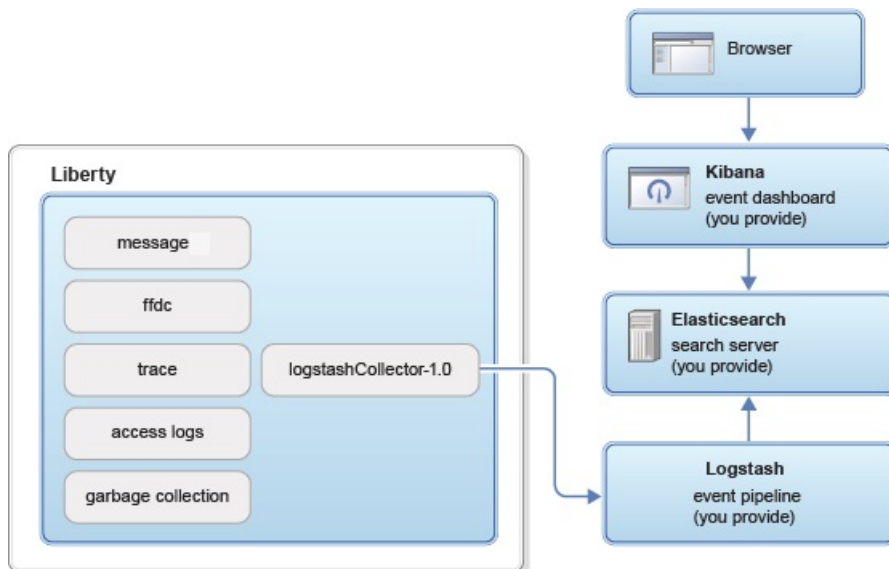


Figure 20. Logstash collector

## Event structure

The collectors send each event as a set of field name-value pairs. Each different type of event has its own set of fields. Knowing which fields each event has is useful when creating your own Kibana dashboards.

The following fields are common and present in all events:

type – a string that identifies the type of event

datetime – time at which the event occurred

hostName – host name of the server that was the source of the event

wlpUserDir – user directory of the server that was the source of the event, for example, D:\wlp\usr

serverName – server name of the server that was the source of the event

sequence – sequence number of event (useful for sorting records with the same time stamp)

Besides the common fields, each of the event types also has its own unique fields:

**Message events (type:"liberty\_message")**

severity – 1 letter severity indicator (F = Fatal, E = Error, W = Warning, A = Audit, I = Info, O = SystemOut, R = SystemErr)

messageId – message ID in the log line, which can be used to find out specific types of errors, for example, SRVE0250I

methodName – method name from log record

className – class name from log record

loggerName – logger name from log record

threadId – thread ID in the log line, for example, 00000015. Note that the thread ID is a string and not a number

message – the message, starting with the message ID

**Trace events (type:"liberty\_trace")**

severity – 1 letter severity indicator (1 = Fine, 2 = Finer, 3 = Finest, > = Entry, < = Exit)

methodName – method name from log record

className – class name from log record

loggerName – logger name from log record

threadId – thread ID in the log line, for example, 00000015. Note that the thread ID is a string and not a number

message – the message

**HTTP access log events (type:"liberty\_accesslog")**

uriPath – Path information for the requested URL. This does NOT contain the query parameters, for example, /pushworksserver/ push/apps/tags

requestMethod – HTTP verb used, for example, GET

remoteHost – remote host IP address in the log line, for example, 127.0.0.1

userAgent – userAgent value in the request.

requestProtocol – Protocol information in the log line, for example, HTTP/1.1

queryString – string representing query string from the HTTP request, for example, color=blue&size=large

bytesReceived – bytes received in the URL, for example, 94 in the sample 1

responseCode – HTTP response code, for example, 200

elapsedTime – time that is taken to serve the request

requestHost – request host IP address in the log line, for example, 127.0.0.1

requestPort – port number of the request

**FFDC events (type:"liberty\_ffdc")**

className – the class that emitted the FFDC entry

exceptionName – the exception that was reported in the FFDC entry

probeID – the unique identifier of the FFDC point within the class

stackTrace – the stack trace of the FFDC incident

objectDetails – the incident details for the FFDC incident

threadId – the thread ID of the ffdc incident

## Garbage Collection events (type:"liberty\_gc")

heap – the total heap available

usedHeap – the amount of heap used

duration – the duration for which GarbageCollection was run

gcType – the type of garbage collection event (for example: Nursery, Global, etc.)

reason – the reason for garbage collection

## Using the Logstash collector

8.5.5.9

Use the Logstash collector feature in Liberty to collect log and other events from your Liberty servers and send them to a remote Logstash server. The collected events can be used for log analysis and troubleshooting purposes.

### Before you begin

The logstashCollector-1.0 feature was tested with Logstash V2.x, Elasticsearch V2.x, and Kibana V4.x. You can use the logstashCollector-1.0 feature with a Logstash server that runs with any of the available output plug-ins from Logstash. However, many users choose to use Logstash V2.x with Elasticsearch V2.x and Kibana V4.x to provide a complete log consolidation and analysis facility. For more information, see Elasticsearch.

### Procedure

1. Set up Logstash V2.x by following the instructions from Elasticsearch.
2. Create or acquire certificate and key pair files for SSL for Logstash. The following example is the command for openssl that can be used for generating a certificate and key pair. Customize the number of days the keys are valid as required.

```
openssl req -x509 -newkey rsa:2048 -keyout logstash.key -out logstash.crt -days 365 -nodes
```

3. For Logstash V2.x and Elasticsearch users, copy the sample into a `liberty_logstash_template.json` file. See the repository for a sample Logstash index template. Customize the `_ttl` defaults as required to indicate the number of milliseconds to keep records of each event type.
4. For Logstash V2.x and Elasticsearch users, copy the sample into a `liberty_logstash.conf` file. See the repository for sample Logstash filters. Customize `lumberjack ssl_certificate` path, `ssl_key` path, and port number as required. Customize Elasticsearch hosts and template path as required.

5. Complete the following steps for each of the Liberty servers that you want to collect events from:

- a. Acquire or create a keystore for the Liberty server. To create a self-signed certificate use the following command. Customize the server name, password, and subject as required.

```
d:\wlp\bin\securityUtility createSSLCertificate --server=myServerName --password="Liberty" --subject=CN=myHostname,OU=defaultServer,C
```

- b. Import the `logstash.crt` file from step 2 into your server's `key.jks` file. Customize the `wlp_install_dir` and server name as required. When prompted for a password, use the certificate password from step 5a.

```
d:\java\bin\keytool -import -noprompt -alias logstash -file logstash.crt -keystore wlp_install_dir\usr\servers\myServerName\resources
```

- c. Run the following command to install the logstashcollector-1.0 feature:

```
d:\wlp\bin\installUtility install logstashcollector-1.0
```

- d. Configure Logstash collector in the `server.xml` file in Liberty by adding the following content. Customize the logstashCollector list of sources, host name, and port as required.

```
<featureManager>
 <feature>logstashCollector-1.0</feature>
</featureManager>
```

```
<keyStore id="defaultKeyStore" password="Liberty" />
```

```

<ssl id="mySSLConfig" trustStoreRef="defaultKeyStore" keyStoreRef="defaultKeyStore" />

<logstashCollector
 source="message,trace,garbageCollection,ffdc,accessLog"
 hostName="localhost"
 port="5043"
 sslRef="mySSLConfig"
/>

```

**Note:** Trace and access logs are usually high volume logs and require more network, CPU, and storage resources to collect.

6. For users of Elasticsearch and Kibana V4.x, import the Kibana dashboard as follows:
  - a. Save the Kibana dashboard JSON to a file on your local file system. For Elasticsearch and Kibana V4.x users, see the repository for a sample Kibana dashboard.
  - b. Import the dashboard into Kibana by clicking **Settings > Objects > Import..** When prompted provide the path to the file you saved in the previous step.
7. Save the dashboard using the save (disk) icon. Enter "Liberty" in the text box that is provided and click the save icon near the text box. The next time that you visit Kibana from any browser you can reload this dashboard using the load icon and clicking **Liberty**.

## Results

Your Liberty servers are configured to send events to your Logstash server, and you can now view your events in the Liberty dashboard using Kibana.

---

## Logging and Trace

The product has a unified logging component that handles messages that are written by the product and provides First Failure Data Capture (FFDC) services.

Additionally, the logging component captures messages that are written to System.out, System.err, java.util.logging, and OSGi logging. The logging component unifies the handling of these messages with other messages written by the product. The logging component is not capable of capturing messages that are written directly by the JVM process, such as `-verbose:gc` output.

There are three primary log files for a server:

1. `console.log` - containing the redirected standard output and standard error from the JVM process. This console output is intended for direct human consumption. The console output contains major events and errors if you use the default `consoleLogLevel` configuration. The console output also contains any messages that are written to the System.out and System.err streams if you use the default `copySystemStreams` configuration. The console output always contains messages that are written directly by the JVM process, such as `-verbose:gc` output. This file is created only if the **server start** command is used, and its location can be altered only by using the `LOG_DIR` environment variable. For more information, see "Administering Liberty from the command line" on page 949.
2. `messages.log` - containing all messages except trace messages that are written or captured by the logging component. All messages that are written to this file contain additional information such as the message time stamp and the ID of the thread that wrote the message. This file does not contain messages that are written directly by the JVM process.
3. `trace.log` - containing all messages that are written or captured by the product. This file is created only if you enable additional trace. This file does not contain messages that are written directly by the JVM process.

## Logging configuration

The logging component can be controlled through the server configuration. The primary location for the logging configuration is in the `server.xml` file. Occasionally, you might need to configure trace to

diagnose a problem that occurs before the server.xml file is processed. In this case, the equivalent configuration properties can be specified in the bootstrap.properties file. If a configuration property is specified in both the bootstrap.properties file and the server.xml file, the value in bootstrap.properties is used until the server.xml file is processed. Then, the value in the server.xml file is used. Avoid specifying different values for the same configuration property in both the bootstrap.properties and the server.xml file.

Table 104. Logging properties for Liberty. Column 1 contains attributes that can be set in the server.xml file. Column 2 contains equivalent properties that can be used in the bootstrap.properties file. Column 3 provides a description of each logging property.

Attribute	Equivalent property	Description
<b>logDirectory</b>	com.ibm.ws.logging.log.directory	This attribute sets the directory for all log files, excluding the console.log file, but including FFDC. By default, <b>logDirectory</b> is set to the LOG_DIR environment variable. The default LOG_DIR environment variable path is <i>WLP_OUTPUT_DIR/serverName/logs</i> . <b>Avoid trouble:</b> Use the LOG_DIR environment variable or com.ibm.ws.logging.log.directory property rather than the <b>logDirectory</b> attribute to configure the directory in which you want all the messages to be written. Otherwise, a few messages are written initially in the logs directory by default, and then the remaining messages are written to the specified directory based on your configuration. The <b>logDirectory</b> attribute might be used to dynamically update the logs to the specified directory while the server is running.
<b>maxFileSize</b>	com.ibm.ws.logging.max.file.size	The maximum size (in MB) that a log file can reach before it is rolled. The Liberty runtime does only size-based log rolling. To disable this attribute, set the value to 0. The maximum file size is approximate. By default, the value is 20. <b>Note:</b> <b>maxFileSize</b> does not apply to the console.log file.
<b>maxFiles</b>	com.ibm.ws.logging.max.files	If an enforced maximum file size exists, this setting is used to determine how many of each of the log files are kept. This setting also applies to the number of exception logs that summarize exceptions that occurred on any particular day. So if this number is 10, you might have 10 message logs, 10 trace logs, and 10 exception summaries in the ffdc/ directory. By default, the value is 2. <b>Note:</b> <b>maxFiles</b> does not apply to the console.log file.
<b>consoleLogLevel</b>	com.ibm.ws.logging.console.log.level	This filter controls the granularity of messages that go to the console.log file. The valid values are INFO, AUDIT, WARNING, ERROR, and OFF. By default, the level is AUDIT. <b>Note:</b> <b>Distributed operating systems</b> Before you change this value, consider the information in section “Unable to interact with the Liberty server after modifying the console log level settings” in the topic “Developer Tools known issues and restrictions” on page 1488.

Table 104. Logging properties for Liberty (continued). Column 1 contains attributes that can be set in the server.xml file. Column 2 contains equivalent properties that can be used in the bootstrap.properties file. Column 3 provides a description of each logging property.

Attribute	Equivalent property	Description
<b>copySystemStreams</b>	com.ibm.ws.logging. copy.system.streams	If true, messages that are written to the System.out and System.err streams are copied to console.log. If false, those messages are written to configured logs such as messages.log or trace.log, but they are not copied to console.log. The default value is true.
<b>messageFileName</b>	com.ibm.ws.logging. .message.file.name	The message log has a default name of messages.log. This file always exists, and contains INFO and other (AUDIT, WARNING, ERROR, FAILURE) messages in addition to System.out and System.err. This log also contains time stamps and the issuing thread ID. If the log file is rolled over, the names of earlier log files have the format messages_timestamp.log
<b>suppressSensitiveTrace</b>		The server trace can expose sensitive data when it traces untyped data, such as bytes received over a network connection. This attribute, when set to true, prevents potentially sensitive information from being exposed in log and trace files. The default value is false.
<b>traceFileName</b>	com.ibm.ws.logging. .trace.file.name	The trace.log file is only created if additional or detailed trace is enabled. <b>stdout</b> is recognized as a special value, and causes trace to be directed to the original standard out stream.
<b>traceSpecification</b>	com.ibm.ws.logging. .trace.specification	The trace string is used to selectively enable trace. The default is *=info.
<b>traceFormat</b>	com.ibm.ws.logging. .trace.format	This attribute controls the format of the trace log. The default format for Liberty is ENHANCED. You can also use BASIC and ADVANCED formats as in the traditional.
8.5.5.4 <b>hideMessage</b>	com.ibm.ws.logging.hideMessage	You can use the <b>hideMessage</b> attribute to configure the messages that you want to hide from the console.log and message.log files. If the messages are configured to be hidden, then they are redirected to the trace.log file.  <b>Note:</b> <b>Distributed operating systems</b> Before you use this attribute, consider the information that is given under the <i>Unable to recognize the start of the server when the hideMessage attribute is used to suppress the messages</i> section in the “Developer Tools known issues and restrictions” on page 1488 topic.

You can set logging properties in the server configuration file by selecting **Logging and Tracing** in the **Server Configuration** view in the developer tools, or by adding a logging element to the server configuration file as follows:

```
<logging traceSpecification="*=audit:com.myco.mypackage.*=finest"/>
```

The format of the log detail level specification is:

<component> = <level>

where <component> is the component for which to set a log detail level, and <level> is one of the valid logger levels (off, fatal, severe, warning, audit, info, config, detail, fine, finer, finest, all). Separate multiple log detail level specifications with colons (:).

**Attention:** For a given logger, the level is determined by the most specific trace specification that applies to that logger.

Components correspond to Java packages and classes, or to collections of Java packages. Use an asterisk (\*) as a wildcard to indicate components that include all the classes in all the packages that are contained by the specified component. For example:

- \* Specifies all traceable code that is running in the application server, including the product system code and customer code.

**com.ibm.ws.\***

Specifies all classes with the package name beginning with com.ibm.ws.

**com.ibm.ws.classloader.JarClassLoader**

Specifies the JarClassLoader class only.

Table 105. Valid logging levels. The following table lists the valid levels for application servers at WebSphere Application Server Version 6 and later.

Version 6 and later logging level	Content / Significance
off	Logging is turned off.
fatal	Task cannot continue and component, application, and server cannot function.
severe	Task cannot continue but component, application, and server can still function. This level can also indicate an impending unrecoverable error.
warning	Potential error or impending error. This level can also indicate a progressive failure (for example, the potential leaking of resources).
audit	Significant event that affects server state or resources
info	General information that outlines overall task progress
config	Configuration change or status
detail	General information that details subtask progress
fine	Trace information - General trace + method entry, exit, and return values
finer	Trace information - Detailed trace
finest	Trace information - A more detailed trace that includes all the detail that is needed to debug problems
all	All events are logged. If you create custom levels, <i>all</i> includes those levels, and can provide a more detailed trace than finest.

The console.log file does not have the same level of management as other log files. The only property that you can change is **consoleLogLevel**. If you are concerned about the increasing size of the console.log file, you can disable the console.log file and use the message log file instead. The same data, in a different format, is written to the message log file, and you can control the size and number of message log files by using the **maxFileSize** and **maxFiles** attributes. For example, the following bootstrap.properties file results in an empty console.log file and a maximum of three rolling 1 MB



loggingMessages.log files. However, messages from the underlying JVM can still be written to the console.log. Settings in the bootstrap.properties file take effect before the message log file is created, so the message log file is initially created as loggingMessages.log and not the default messages.log.

```
com.ibm.ws.logging.max.file.size=1
com.ibm.ws.logging.max.files=3
com.ibm.ws.logging.console.log.level=OFF
com.ibm.ws.logging.message.file.name=loggingMessages.log
```

The console.log file is reset when the server is restarted.

**Note:** On all platforms, logs are written in the default system encoding.

- **Windows** On Windows systems, there are two types of encoding: OEM code page, which is used for console output, and ANSI code page, which is used to read and write files. The console.log file uses the OEM code page, and all other logs use the ANSI code page.
- **Distributed operating systems** On all other platforms, all log files use the default encoding.

**Note:** For general help on understanding message formats, see Message log interpretation.

---

## Viewing trace and message log files by using developer tools

Linux

Windows

8.5.5.7

You can view trace and message log files for either local or remote servers by using WebSphere Developer Tools.

### Before you begin

- To view the trace file, you must enable tracing by specifying the traceformat and tracespecification settings in the server.xml file or the bootstrap.properties file. For more information about enabling tracing, see Liberty: Logging and Trace.
- You can change the default log file names from messages.log and trace.log, and the default directory name from WLP\_OUTPUT\_DIR/serverName/logs by changing any of the configuration files.
  - Configuration files include server.xml, bootstrap.properties, and server.properties.
- Actions for local servers:
  - Ensure that the log file is in the log directory that you specified. Otherwise, the menu actions for the local server are disabled.
  - If you change the bootstrap.properties file or the server.env file, you must restart the server for changes to take effect. Menu actions are temporarily disabled until the server is restarted.
  - If the same configuration setting is specified in more than one configuration file, the setting in the server.xml file overrides the other values.
- Actions for remote servers:
  - Ensure that the server is running. If it is not running, menu actions are not displayed.
  - If you change the default location of a log file, include remote access by adding either a ReadDir or a WriteDir configuration to your server.xml file to indicate the log location. The following example shows how to add remote access.

```
<logging logDirectory="${server.output.dir}/log_a" maxFiles="10" traceFormat="BASIC" traceSpecification="com.i
<remoteFileAccess>
 <writeDir >${wlp.user.dir}</writeDir>
 <writeDir>${server.config.dir}</writeDir>
 <writeDir>${server.output.dir}</writeDir>
 <writeDir>${server.output.dir}/log_a</writeDir>
</remoteFileAccess>
```

## Procedure

1. To view a list of servers, select **Window > Show View > Servers**.
2. Right-click a server.
  - To view the trace log file, select **Open Log Files > Trace File** from the menu.
  - To view the messages log file, select **Open Log Files > Message Log File** from the menu.

For remote servers, downloading log files can take several seconds depending on the size of the log file.

## Results

The log file that you selected is displayed in the default editor. The following example shows records in a message log file.

```
[3/3/11 23:01:30:147 EST] 0000000f ApplicationMg Z WSVR0221I: Application started: DefaultApplication
```

In this example, *[3/3/11 23:01:30:147 EST]* is the time stamp, *0000000f* is the thread ID, *ApplicationMg* is the logger, *Z* is the level, *WSVR0221I* is the message ID, and *Application started: DefaultApplication* is the message.

---

## Timed operations and JDBC calls

Timed operations generate a logged warning when JDBC calls in the application server are operating more slowly or quickly than expected.

### Overview

When enabled, the timed operation feature tracks the duration of JDBC operations running in the application server. In cases where operations take more or less time to execute than expected, the timed operation feature logs a warning. Periodically, the timed operation feature will create a report, in the application server log, detailing which operations took longest to execute. If you run the server dump command, the timed operation feature will generate a report containing information about all operations it has tracked. You can use the information listed in these reports to decide if anything is running slower or faster than you expect.

Periodically, the system generates a report to the logs that contains the ten longest JDBC timed operations. The frequency and enablement of this report is configurable in the server.xml file, with a default of once per day (24 hours).

To enable timed operations, add the `timedOperations-1.0` feature to the `server.xml` file.

You can disable the generation of the report to the logs, or change the frequency of the report, for example to once every 12 hours, using the `timedOperation` element as shown in the following example:

```
<timedOperation enableReport="false" reportFrequency="12" />
```

You can also use the `maxNumberTimedOperations` attribute to log a warning when the total number of timed operations reaches the value specified by this attribute. The number of timed operations is monitored and useful to know since each timed operation is allocated memory from the heap, and if you find that the number of timed operations is excessive, you can disable the timed operations feature. You can use the following example to configure the `maxNumberTimedOperations` attribute:

```
<timedOperation enableReport="false" reportFrequency="12" maxNumberTimedOperations="10000"/>
```

This example results in a warning message in the log as follows when the number of timed operations exceeds 10000:

[4/18/13 23:01:37:316 EDT] 0000002c com.ibm.wsspi.timedoperations.TimedOperationService W TRAS0094I: The total number of timed operations is 10000, which exceeds the configured maximum number of 10000. You can also find the number of timed operations in the report that is periodically generated to the logs. If you find that the number of timed operations is excessive, you can disable the timed operations feature.

### 8.5.5.2

If you set the value of the *com.ibm.timedOperations.autoCleanup* WebSphere environment variable to true, the server automatically limits the number of tracked timed operations to the value specified in the <maxNumberTimedOperations> attribute. A warning is logged when the total number of timed operations reaches the maximum value specified. To limit the number of tracked timed operations, when a new timed operation is required to be tracked, the least recently used timed operation record is deleted. When the number of timed operations that are tracked reaches the specified maximum value, a warning message is displayed as follows:

TRAS0095I: The total number of timed operations has reached the configured maximum of 10000. As new timed operations are

You can also use the **server dump** command to get a full report of all timed operations in the `messages.log` file, grouped by type, and sorted within each group by average of actual duration.

The following example shows a sample logged message:

```
[3/14/13 14:01:25:960 CDT] 00000025 TimedOperatio W TRAS0080W: Operation websphere.datasourc.execute:
jdbc/exampleDS:insert into cities values ('myHomeCity', 106769, 'myHomeCountry') took 1.541 ms to complete,
which was longer than the expected duration of 0.213 ms based on past observations.
```

The following example shows a sample automatically generated report in the log:

```
[12/13/12 7:42:29:509 CST] 0000001d com.ibm.wsspi.timedoperations.TimedOperationService I TRAS0092I:
The following operations took the longest time to run since the last report has been generated:
Operation websphere.datasourc.execute:jdbc/exampleDS:insert into cities values ('myHomeCity',
106769, 'myHomeCountry') took 194ms to complete
Operation websphere.datasourc.execute:jdbc/exampleDS:select county from cities where name=
'myHomeCity' took 187ms to complete
Operation websphere.datasourc.execute:jdbc/exampleDS:drop table cities took 182ms to
complete\Operation websphere.datasourc.execute:jdbc/exampleDS:insert into cities values
('myHomeCity', 106769, 'myHomeCountry') took 151ms to complete
```

For the full timed operation configuration reference, see the `timedOperation` element in the following topic: **\*\*\* MISSING FILE \*\*\***

---

## Event Logging



### 8.5.5.6

As part of the monitoring and diagnostic capabilities, the WebSphere Application Server Liberty generates events at various components of Java Platform, Enterprise Edition to track the requests. The `eventLogging-1.0` feature logs such events when the application requests are running. Using this feature, the user can track the requests that are running in the WebSphere Application Server Liberty. Each request is associated with a unique correlator called the request ID and the context information that helps the user to understand the request-specific data.

The event logging feature is controlled through the server configuration. The feature is configured in the `server.xml` file.

The following sample log shows the end-to-end event logs for the `AAY6Ta1VDTO_AAAAAAAAAAAK` request ID and `TradeWeb` context:

```
[12/15/14 18:24:29:528 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=webspher
[12/15/14 18:24:29:531 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=webspher
[12/15/14 18:24:29:532 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=webspher
[12/15/14 18:24:29:533 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=webspher
[12/15/14 18:24:29:534 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=webspher
[12/15/14 18:24:29:547 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.
[12/15/14 18:24:29:556 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.
```

```

[12/15/14 18:24:29:671 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.
[12/15/14 18:24:29:672 IST] 0000002e EventLogging I BEGIN requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.
[12/15/14 18:24:29:677 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.da
[12/15/14 18:24:29:684 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.se
[12/15/14 18:24:29:685 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.se
[12/15/14 18:24:29:686 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.se
[12/15/14 18:24:29:687 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.se

```

The request begins at the BEGIN websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet" event (refer to the first line in the sample code) and ends at END websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet" (refer to the last line in the sample code). The total time that is taken by this request is also displayed at the end (which is 158.283 ms in the sample code).

You can look at the child requests by looking at the BEGIN and END of the main request. You can also find the time that is taken by each of the child request.

For best performance, you might use binary logging when event logging is enabled. The eventType, contextInfo, and requestID attributes from the event log entries are stored as log record extensions. You can use those log record extensions to filter the logs with the **binaryLog** command.

## Parsing the event log entries in the messages.log file

The event logs capture the information of the events in the following format:

```
[Log mode] [Request Identifier] # [Event Type] # [Context Information] # [Duration] (optional)
```

where

- Log mode indicates whether the log was recorded at the entry to the event or the exit from the event. BEGIN refers to the entry and END refers to the exit.
- Request identifier is a unique string that is assigned to each request. This can be used for filtering events that belong to a specific request.
- Event type provides information about the event source and can be any of the supported event types as given in Table 106.
- Context information of the event provides details relevant to the event type. The information varies depending on the event type.
- Duration indicates the time that is taken by the event. The duration appears only in the exit event entries. Example: 158.283 ms.

Except the log mode, which is separated by a space, all other log attributes are separated by #(space#space). For example,

```
[12/15/14 18:24:29:687 IST] 0000002e EventLogging I END requestID=AAY6Ta1VDTO_AAAAAAAAAAAK # eventType=websphere.servlet.service
```

The following table lists the event types that are supported by event logging:

Table 106. Supported event types along with the relevant context information

Component	Event types	Context information	Example
Servlet	websphere.servlet.destroy websphere.servlet.service	[Application Name]   [Servlet Name]	contextInfo=TradeWeb   displayName=Qte.js
Session	websphere.session.dbSessionDestroyed websphere.session.dbSessionDestroyed websphere.session.sessionAccessed websphere.session.sessionCreated websphere.session.sessionDestroyedByTimeout websphere.session.sessionDestroyed websphere.session.sessionLiveCountDec websphere.session.sessionLiveCountInc websphere.session.sessionReleased  websphere.session.getAttribute websphere.session.setAttribute	[Session ID]   [Session Attribute Name]	contextInfo=EuitabHZU0D7J2u01HDdAG0   user=...

Table 106. Supported event types along with the relevant context information (continued)

Component	Event types	Context information	Example
JDBC	websphere.datasources.execute websphere.datasources.executeQuery websphere.datasources.executeUpdate websphere.datasources.psExecute websphere.datasources.psExecuteQuery websphere.datasources.psExecuteUpdate websphere.datasources.rsCancelRowUpdates websphere.datasources.rsDeleteRow websphere.datasources.rsInsertRow websphere.datasources.rsUpdateRow	Indi Name Of Data Source   conf {SQL in jdbc/TradeDataSource   sele	

## Slow and hung request detection



8.5.5.6

The requestTiming-1.0 feature provides diagnostic information when the duration of any request exceeds the configured threshold.

The request timing feature can track the duration of every request that is coming into the system. You can configure the feature to watch for slow and hung requests.

- “Slow request detection”
- “Hung request detection” on page 1470

### Slow request detection

When a request has been running for longer than configured, a warning message is written in the messages log file. Details about the request and events that made up the request are captured.

The following sample shows the log message for a request that crossed the slow request threshold (default is 10 seconds):

```
[12/1/14 11:58:09:629 IST] 0000001d com.ibm.ws.request.timing.SlowRequestTimer W TRAS0112W: Req
 at java.util.HashMap.getEntry(HashMap.java:516)
 at java.util.HashMap.get(HashMap.java:504)
 at org.apache.derby.iapi.store.access.BackingStoreHashtable.get(Unknown Source)
 at org.apache.derby.impl.sql.execute.HashScanResultSet.getNextRowCore(Unknown Source)
 at org.apache.derby.impl.sql.execute.NestedLoopJoinResultSet.getNextRowCore(Unknown Source)
 at org.apache.derby.impl.sql.execute.ProjectRestrictResultSet.getNextRowCore(Unknown Source)
 at org.apache.derby.impl.sql.execute.DMLWriteResultSet.getNextRowCore(Unknown Source)
 at org.apache.derby.impl.sql.execute.DeleteResultSet.setup(Unknown Source)
 at org.apache.derby.impl.sql.execute.DeleteResultSet.open(Unknown Source)
 at org.apache.derby.impl.sql.GenericPreparedStatement.executeStmt(Unknown Source)
 at org.apache.derby.impl.sql.GenericPreparedStatement.execute(Unknown Source)
 at org.apache.derby.impl.jdbc.EmbedStatement.executeStatement(Unknown Source)
 at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeStatement(Unknown Source)
 at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeUpdate(Unknown Source)
 at com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.executeUpdate(WSJdbcPreparedStatement.java:626)
 at com.ibm.websphere.samples.trade.direct.TradeDirect.resetTrade(TradeDirect.java:1832)
 at com.ibm.websphere.samples.trade.web.TradeConfigServlet.doResetTrade(TradeConfigServlet.java:65)
 at com.ibm.websphere.samples.trade.web.TradeConfigServlet.service(TradeConfigServlet.java:348)
 at javax.servlet.http.HttpServlet.service(HttpServlet.java:668)
 at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1275)

 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1121)
```

```
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:614)
at java.lang.Thread.run(Thread.java:769)
```

Duration	Operation
10007.571ms +	websphere.servlet.service   TradeWeb   TradeConfigServlet?action=resetTrade
3.923ms	websphere.datasource.psExecuteUpdate   jdbc/TradeDataSource   delete from holdingejb where holding_
0.853ms	websphere.datasource.psExecuteUpdate   jdbc/TradeDataSource   delete from accountprofileejb where
5271.341ms +	websphere.datasource.psExecuteUpdate   jdbc/TradeDataSource   delete from orderejb where account_

The request continues to be monitored, and a further warning is logged if the request is running beyond another 10 seconds. The log messages are in the following format:

```
TRAS0112W: Request <(Request ID)> has been running on thread <THREADID> for at least <DURATION>. The following stack
<STACK TRACE>
<DURATION AND OPERATIONS Table>
```

### REQUEST ID

This same ID can be used to search for log and trace messages corresponding to the request. In particular, if you use binary logging, you can search for log and trace entries with the same requestID extension by using the binary log command.

### STACK TRACE

Indicates the method that is running. In the previous sample, you can see a stack trace of the current request after TRAS0112W line.

### DURATION AND OPERATIONS Table

After the stack trace, you can find the tabular format of the request that shows the duration and operation (also referred to as the event). The Duration column indicates the time that is taken by the corresponding operation of the request. The plus sign (+) indicates events within the request that are still running. The next line shows the duration without +, which indicates that the corresponding operation has completed in the specified duration. Operation shows the EVENT TYPE and CONTEXT INFO (which is optional) for that operation. For more information about event types and context information, see “Event Logging” on page 1467.

By analyzing the messages, you can figure out why the request is slow. However, it might be difficult to determine whether the request is stuck at that point or is still running slowly. Hence you can see three messages that are logged for any slow request at the interval of specified *<slowRequestThreshold>*. Using the three different stack trace and request data, you get a better insight into the issue. After the third warning, no further warnings are logged about the request unless the duration of the request crosses the hung request detection threshold.

## Hung request detection

If the request exceeds the default hungRequestThreshold or the configured threshold value, a warning message is written in the messages log file along with the details about the request. Details about the request and events that made up the request are captured. In case of hung request detection, a series of three thread dumps (javacores) is taken, with 1 minute delay between them. The following log message sample shows the log messages for a request that crossed the hung request detection threshold. The default duration value is 10 min. The value that is configured in the following example is 4 min.

```
[WARNING] TRAS0114W: Request AAA7WlpP717_AAAAAAAAAAAAA has been running on thread
00000021 for at least 240001.015ms. The following table shows the events that have run during this
request.
```

Duration	Operation
240001.754ms +	websphere.servlet.service   TestWebApp   TestServlet?sleepTime=480000
0.095ms	websphere.session.setAttribute   mCzBMyzMvAEnjMJjx9zQYIw   userID
0.007ms	websphere.session.setAttribute   mCzBMyzMvAEnjMJjx9zQYIw   visitCount

If a request gets completed later, which was detected to be hanging initially, a message similar to the following example is logged: TRAS0115W: Request AAA7WlpP7I7\_AAAAAAAAAAAAA, which was previously detected to be hung, has completed after 479999.681 s.

**Note:** When a request is detected to be hanging, a series of three thread dumps is initiated. After the completion of the three thread dumps, further thread dumps are created only if the new requests are detected to be hanging.

---

## Binary logging

Binary logging is a high performance log and trace facility based on the High Performance Extensible Logging (HPEL) technology in WebSphere Application Server traditional.

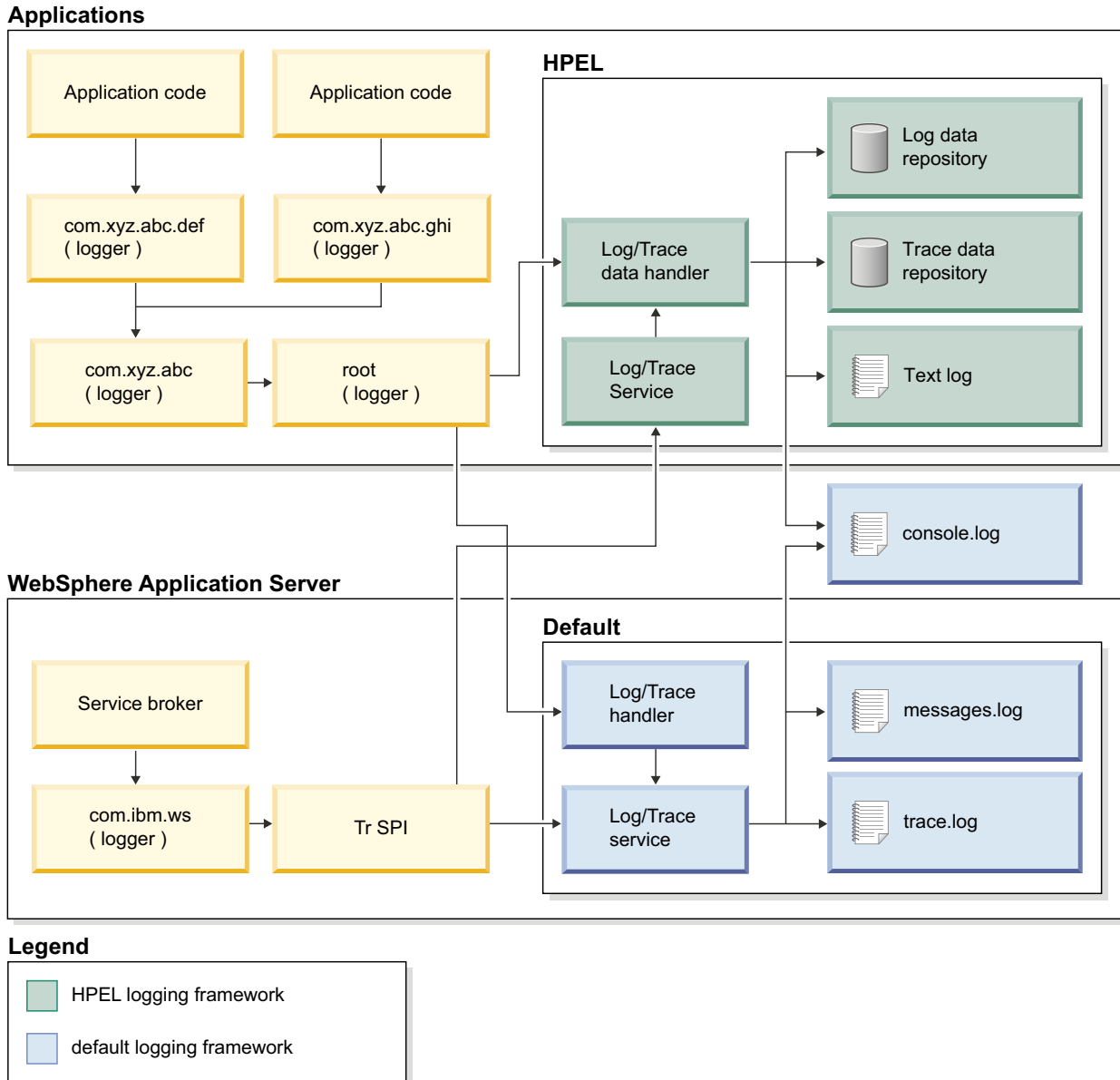
### Overview

**Note:** You must enable the binary logging facility to use it.

Binary logging provides a convenient mechanism for storing and accessing log, trace, System.err, and System.out information produced by the application server or your applications. It is an alternative to the default log and trace facility, which provides the JVM logs and diagnostic trace files commonly named messages.log and trace.log.

### Log and trace storage

Binary logging provides a log data repository and a trace data repository. See the following figure to understand how applications and the application server store log and trace information.



### Log data repository

The log data repository is a storage facility for log records. Log data is typically intended to be reviewed by administrators. This includes any information applications or the server write to System.out, System.err, OSGi logging service at level LOG\_INFO or higher (including LOG\_INFO, LOG\_WARNING, and LOG\_ERROR), or java.util.logging at level Detail or higher (including Detail, Config, Info, Audit, Warning, Severe, Fatal, and any custom levels at level Detail or higher).

### Trace data repository

The trace data repository is a storage facility for trace records. Trace data is typically intended for use by application programmers or by the WebSphere Application Server support team. This includes any information applications or the server write to the OSGi logging service at level LOG\_DEBUG or java.util.logging at levels below level Detail (including Fine, Finer, Finest, and any custom levels below level Detail).



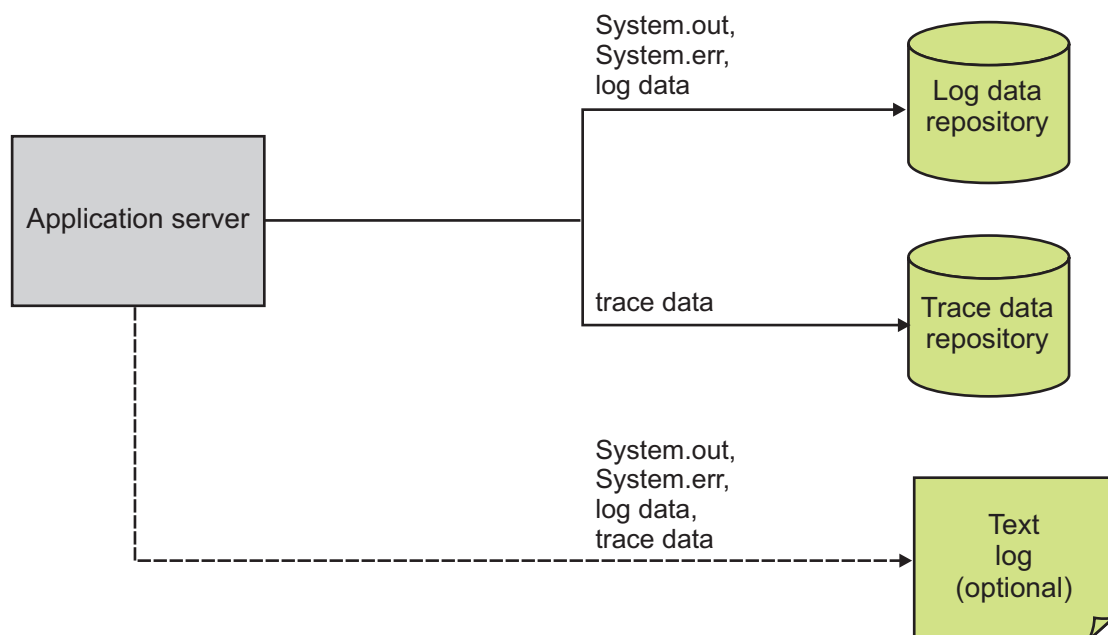
## Log and trace performance

Binary logging has been designed and tested to significantly outperform the default log and trace facility. One result is that the application server can run with trace enabled while causing less impact to performance than tracing the same components using the default log and trace framework. Another result is that applications that frequently write to the logs can run faster when using binary logging.

### Log and trace events are each stored in only one place

Log events, System.out, and System.err are stored in the log data repository. Trace events are stored in the trace data repository. Storing each type of event in only one place ensures that performance is not wasted on redundant data storage.

**Note:** The console log should be disabled in cases where logging performance is important. Any content written to the console log will already be stored in the log data repository.



### Data is not formatted unless it is needed

Formatting data for a user to read uses processor time. Rather than format log event and trace event data at run time, log and trace data are stored more rapidly in a proprietary binary representation. This improves the performance of the log and trace facility. By deferring log and trace formatting until the `binaryLog` command is run, sections of the log or trace that are never viewed are never formatted.

### Log and trace data are buffered before being written to disk

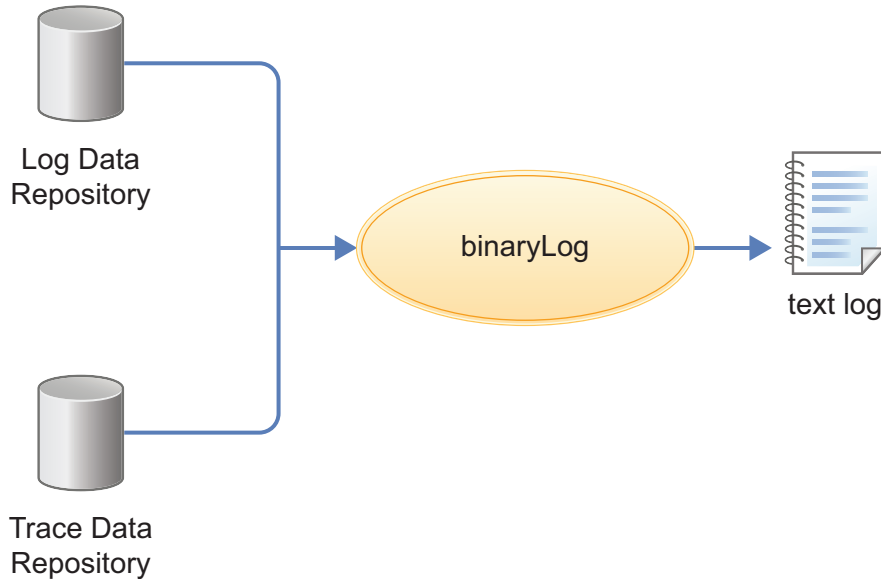
Writing large blocks of data to a disk is more efficient than writing the same amount of data in small blocks. The binary logging facility provides the capability to buffer log and trace data before writing it to disk. By default, log and trace data are stored in an 8 KB buffer before being written to disk. If the buffer is filled within 10 seconds, the buffer is written to disk. If the buffer is not filled within that time it is automatically written to disk to ensure that the logs have the most current information.

## Administration of log and trace

Binary logging has been designed to be easy to configure and understand. For example, administrators can easily configure how much disk space to dedicate to logs or trace, or how long to retain log and trace records, and leave the management of log and trace content up to the server. As another example all log, trace, System.out, and System.err content can be accessed using one easy-to-use command (`binaryLog`), avoiding any possible confusion over which file to access for certain content.

### Reading from the log data and trace data repositories

The log data and trace data repositories are stored in a WebSphere Application Server proprietary format and cannot be read using text file editors such as Notepad or VI. You can copy the log data and trace data repositories in to a plain text format using the `binaryLog` command.



### `binaryLog` command

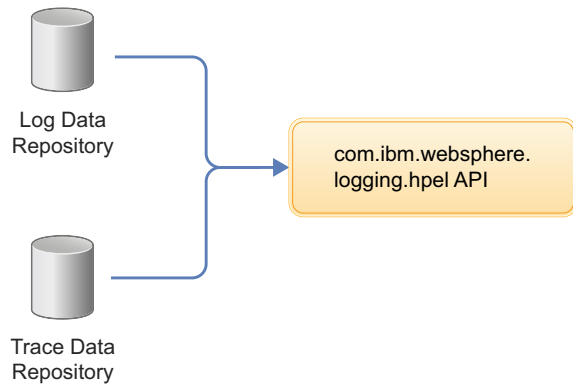
`binaryLog` is an easy-to-use command-line tool provided for users to work with the log data and trace data repositories. `binaryLog` provides filtering and formatting options that make finding important content in the log data and trace data repositories easy. For example, a user might filter any errors or warnings, then filter all log and trace entries that occurred within 10 seconds of a key error message on the same thread.

### Filtering using log and trace record extension content

The binary logging facility provides the capability for developers to add custom extensions to log and trace records using a log record context API (`com.ibm.websphere.logging.hpel.LogRecordContext`). You can use the `binaryLog` command-line tool to filter records based on the content of log and trace record extensions.

## Development resources

Binary logging has been designed to make working with log and trace content more flexible and effective than the default logging facility. Log and trace content can be easily filtered to show only the records that are of interest. You can use the command line (see the description of the `binaryLog` command), or developers can create powerful log handling programs using the HPEL API.



### Reading the log data and trace data

An API has been provided to make it easy for developers to develop tools to consume content from the binary log and trace repositories. For example, a developer might write a Java program to search the log and trace content to find any messages with message IDs that match a known list of important message IDs. This API is in the `com.ibm.websphere.logging.hpel` package. Refer to the API documentation for details on the HPEL log reading API.

### Log and trace record extensibility

Developers can add custom extensions to log and trace records through a log record context API (`com.ibm.websphere.logging.hpel.LogRecordContext`). When binary logging stores log and trace records, it includes any extensions present in the log record context on the same thread. For example, a developer might write a servlet filter to add important HTTP request parameters to the log record context. While that servlet runs, HPEL API adds those extensions to any log and trace records created on the same thread.

As with other log and trace record fields, developers can access the record extensions using the HPEL API. This is useful when writing tools to read from log and trace repositories. Developers can also make use of the log record context API to access extensions in custom log handlers, filters, and formatters at run time.

## BinaryLog command options

Use the **binaryLog** command to view or copy the contents of a binary logging repository, or list the available server process instances in the repository. The `binaryLog` command is equivalent to the `logViewer` command in the profile bin directory of the traditional application server.

The binary log and trace facility writes to a repository in a binary format. You can view, query and filter the repository using the `binaryLog` command. The `binaryLog` command provides options for quickly converting repository contents into a text file in various formats, such as basic and advanced formats. The command also provides options to make getting the data you need from the logs easier; for example, allowing you to filter what log records you want by level, logger name, or date and time.

### Syntax

The command syntax is as follows:

```
binaryLog action {serverName | repositoryPath} [options]
```

The value of **options** is different based on the value of **action**.

### Parameters

The following actions are available for the `binaryLog` command:

## view

Read a repository, optionally filter it, and create a version that users can read.

The command syntax is as follows:

```
binaryLog view {serverName | repositoryPath} [options]
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains both the logdata and tracedata directories.

**Note:** If neither a *serverName* nor a *repositoryPath* is specified on the command line, the task is performed against the default server instance, `defaultServer`, if it exists.

Filter options:

All filters are optional. When multiple filters are used, they are logically ANDed together.

- `--minDate=value`  
Filter based on minimum record creation date. Value must be specified as either a date (for example `--minDate="2/20/13"`) or a date and time (for example `--minDate="2/20/13 16:47:21:445 EST"`).
- `--maxDate=value`  
Filter based on maximum record creation date. Value must be specified as either a date (for example `--maxDate="2/20/13"`) or a date and time (for example `--maxDate="2/20/13 16:47:21:445 EST"`).
- `--minLevel=value`  
Filter based on minimum level. Value must be one of `FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`.
- `--maxLevel=value`  
Filter based on maximum level. Value must be one of the following: `FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`.
- `--includeLogger=value[,value]*`  
Include records with specified logger name. Value may include `*` or `?` as a wildcard.
- `--includeMessage=value`  
Filter based on message name. Value may include `*` or `?` as a wildcard.
- `--includeThread=value`  
Include records with specified thread id. Values must be in hexadecimal (for example, `--includeThread=2a`).
- `--includeExtension=name=value[,name=value]*`  
Include records with specified extension name and value. Value may include `*` or `?` as a wildcard. To include a comma in the value, you must use `"\,"`.
- `--includeInstance=value`  
Include records from the specified server instance. Value must either be `"latest"` or be a valid instance ID. Run this command using the `listInstances` action to see a list of valid instance IDs.

Monitor option:

`--monitor`

Continuously monitor the repository and output new content as it is generated.

Output options:

- `--format={basic | advanced | CBE-1.0.1}`

Specify the output format to use. "basic" is the default format.

- `--encoding=value`

Specify the character encoding to use for output.

## copy

Read a repository, optionally filter it, and write the contents to a new repository.

The command syntax is as follows:

```
binaryLog copy {serverName | repositoryPath} targetPath [options]
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains the logdata and tracedata directories.

*targetPath*

Specify the path at which to create a new repository. The *targetPath* must be specified.

**Note:** Either *serverName* or *repositoryPath* must be specified, as well as the *targetPath*.

Filter options:

All filters are optional. When multiple filters are used, they are logically ANDed together.

- `--minDate=value`

Filter based on minimum record creation date. Value must be specified as either a date (for example `--minDate="2/20/13"`) or a date and time (for example `--minDate="2/20/13 16:52:32:808 EST"`).

- `--maxDate=value`

Filter based on maximum record creation date. Value must be specified as either a date (for example `--maxDate="2/20/13"`) or a date and time (for example `--maxDate="2/20/13 16:52:32:808 EST"`).

- `--minLevel=value`

Filter based on minimum level. Value must be one of the following: FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL.

- `--maxLevel=value`

Filter based on maximum level. Value must be one of the following: FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL.

- `--includeLogger=value[,value]*`

Include records with specified logger name. Value may include \* or ? as a wildcard.

- `--excludeLogger=value[,value]*`

Exclude records with specified logger name. Value may include \* or ? as a wildcard.

- `--includeMessage=value`

Filter based on message name. Value may include \* or ? as a wildcard.

- `--includeThread=value`

Include records with specified thread id. Values must be in hexadecimal (for example, `--includeThread=2a`).

- `--includeExtension=name=value[,name=value]*`

Include records with specified extension name and value. Value may include \* or ? as a wildcard. To include a comma in the value, you must use "\",

- `--includeInstance=value`

Include records from the specified server instance. Value must either be "latest" or be a valid instance ID. Run this command using the listInstances action to see a list of valid instance IDs.

## listInstances

List the IDs of server instances in the repository. A server instance is the collection of all log/trace records written from the time a server is started until it is stopped. Server instance IDs can be used with the `--includeInstance` option of the `binaryLog view` action.

The command syntax is as follows:

```
binaryLog listInstances {serverName | repositoryPath}
```

*serverName*

Specify the name of a Liberty server with a repository to read from.

*repositoryPath*

Specify the path to a repository to read from. This is typically the directory that contains the `logdata` and `tracedata` directories.

**Note:** If *serverName* or *repositoryPath* are not specified on the command line, the task is performed against the default server instance, `defaultServer`, if it exists.

Be aware of `binaryLog` filtering optimizations. The `binaryLog` tool is able to filter log and trace data most efficiently when used with the following filter options:

- `--minDate`
- `--maxDate`
- `--includeThread`
- `--minLevel`
- `--maxLevel`

## Example usage

See the following examples of `binaryLog` commands.

- Display all events in the `defaultServer` repository between July 19th, 2013 and August 2nd, 2013.  

```
binaryLog view --minDate=07/19/13 --maxDate=08/02/13
```
- Display new events from server `myServer`, whose specified level is `WARNING` or higher, using the advanced format as the server writes them to the log repository.  

```
binaryLog view myServer --monitor --minLevel=WARNING --format=advanced
```
- Write log messages from a repository at `/apps/server1/logs`; include only those that were written to the error stream of a specific repository.  

```
binaryLog view /apps/server1/logs --includeLogger=SystemErr
```
- View events from the `defaultServer` repository that occurred before September 14th, 2012 4:28 PM eastern daylight time.  

```
binaryLog view --maxDate="09/14/12 16:28:00:000 EDT"
```
- Write events from the `defaultServer` repository that contain a 'thread' extension with value 'Default Executor-thread-4'  

```
binaryLog view --includeExtension=thread="Default Executor-thread-4" --format=advanced
```
- View the list of server instances in the `defaultServer` repository:  

```
binaryLog listInstances
```

Using `D:\wlp\usr\servers\defaultServer\logs` as repository directory.

Instance ID	Start Date
1358809441761	1/21/13 18:04:01:761 EST
1358864476191	1/22/13 9:21:16:191 EST
1358869523192	1/22/13 10:45:23:192 EST
1358871281166	1/22/13 11:14:41:166 EST
1358879829000	1/22/13 13:37:09:000 EST
1358892222067	1/22/13 17:03:42:067 EST
- View events from the `defaultServer` using one of the instance IDs from the previous example:  

```
binaryLog view --includeInstance=1358871281166
```

- Copy events from the defaultServer, whose specified level is WARNING or higher, from the latest server instance to a new repository at d:\toSupport directory.

```
binaryLog copy defaultServer d:\toSupport --minLevel=warning --includeInstance=latest
```

## Configuring binary logging in Liberty

Use this information as a guide for configuring binary logging in your Liberty.

### About this task

Binary logging provides faster log and trace handling capabilities and more flexible ways to use log and trace content than the default Liberty log and trace framework.

A server configuration consists of a bootstrap.properties file, a server.xml file, and any (optional) files that are included with those files. The bootstrap.properties file specifies properties that need to be available before the main configuration is processed, and are kept to a minimum. The server.xml file is the primary configuration file for the server.

The server.xml file and its associated files use a simple xml format that is suitable for most text editors.

**Distributed operating systems** A richer editing experience is provided by the eclipse server adapter for Liberty (WAS4D+ adapter), which uses a generated schema to provide drop-down lists of available choices, auto-completion, and other editing tools. For a description of the eclipse server adapter for Liberty, see “Editing the Liberty configuration by using developer tools” on page 938.

The bootstrap.properties file specifies whether the server uses binary logging as the log and trace framework, or the default log and trace framework. A server restart is required to switch between binary logging and the default log and trace framework.

You can modify the configuration of binary logging through the server configuration or the bootstrap.properties file.

- Server configuration: To get logging from your own code, which is loaded after server configuration processing, use the server configuration to configure binary logging.
- bootstrap.properties file: You might need to set logging properties to take effect before the server configuration files are processed. For example, if you need to analyze problems that occur early in server start or configuration processing. In this case, you can configure binary logging in the bootstrap.properties file.

You can set Logging properties in either the bootstrap.properties or the server.xml file. Use attributes in the server.xml file, or use equivalent properties in the bootstrap.properties file. Any settings in the bootstrap.properties file are used from the time the server reads the bootstrap.properties file until the time the server.xml file is processed. If the logging properties in the bootstrap.properties file are not replaced or reset in the server.xml file, the property values in the bootstrap.properties file continue to be used.

When binary logging is enabled, the **maxFileSize**, **maxFiles**, **messageFileName**, **traceFileName**, and **traceFormat** logging element attributes are ignored (since binary logging runs without trace.log and messages.log files). The **traceSpecification**, **consoleLogLevel**, and **logDirectory** attributes continue to be used to set the trace specification, the level for the console log, and the placement of the log and trace files.

If you set logging or binary logging attributes in the server.xml file, you can avoid changes in configuration between startup time and runtime by setting the corresponding properties in the bootstrap.properties file to the same value. If no logging or binary logging properties are set in the bootstrap.properties file, the server uses the default logging settings.

## Procedure

- Enable binary logging for the server by updating the `bootstrap.properties` file. In the `bootstrap.properties` file, add the following text on a line by itself:  
`websphere.log.provider=binaryLogging-1.0`
- Use the following parameters to configure binary logging. All subelements that are listed are subelements of the logging element in the `server.xml` file. The following table lists the attributes that are configurable in the `server.xml` file and the equivalent properties that can be set in the `bootstrap.properties` file:

Table 107. Binary logging attributes that are configurable in `server.xml` and the equivalent properties that can be set in `bootstrap.properties`

Logging subelement	Attribute	Equivalent <code>bootstrap.properties</code> property
binaryLog	<code>purgeMaxSize</code>	<code>com.ibm.hpel.log.purgeMaxSize</code>
	<code>purgeMinTime</code>	<code>com.ibm.hpel.log.purgeMinTime</code>
	<code>fileSwitchTime</code>	<code>com.ibm.hpel.log.fileSwitchTime</code>
	<code>bufferingEnabled</code>	<code>com.ibm.hpel.log.bufferingEnabled</code>
	<code>outOfSpaceAction</code>	<code>com.ibm.hpel.log.outOfSpaceAction</code>
binaryTrace	<code>purgeMaxSize</code>	<code>com.ibm.hpel.trace.purgeMaxSize</code>
	<code>purgeMinTime</code>	<code>com.ibm.hpel.trace.purgeMinTime</code>
	<code>fileSwitchTime</code>	<code>com.ibm.hpel.trace.fileSwitchTime</code>
	<code>bufferingEnabled</code>	<code>com.ibm.hpel.trace.bufferingEnabled</code>
	<code>outOfSpaceAction</code>	<code>com.ibm.hpel.trace.outOfSpaceAction</code>

The following example shows a `bootstrap.properties` file that is configured to enable binary logging:  
`websphere.log.provider=binaryLogging-1.0`

The following example shows a `server.xml` file with the binary logging subelements. The log content is set to expire after 96 hours and the trace content is set to retain a maximum of 1024MB:

```
<server description="new server">
 <logging>
 <binaryLog purgeMinTime="96"/>
 <binaryTrace purgeMaxSize="1024"/>
 </logging>
</server>
```

For the full logging configuration reference, see the `logging`, `binaryLog`, and `binaryTrace` elements in the `**** MISSING FILE ****`.

## Results

After you restart the server, binary logging is enabled and configured.

---

## Runtime environment known issues and restrictions

There are some known issues and restrictions that apply when working with the Liberty runtime environment.

### Distributed operating systems

See also “Developer Tools known issues and restrictions” on page 1488.



## Minimum supported Java levels

Liberty is supported with any compliant Java SE 6, Java SE 7, or Java SE 8 runtime environment (JRE) or Java SDK, subject to the minimum supported levels shown for the following specific implementations.

### Java SE 6 runtime environment

For the Java SDK from IBM, the minimum supported level is 6.0 (J9 2.6) SR 1. For the JDK from Oracle, the minimum supported level is Java 6 update 26.

### Java SE 7 runtime environment

For the Java SDK from IBM, the minimum supported level is IBM Runtime Environment, Java Technology Edition 7.0.4.1. For the JDK from Oracle on Windows and Linux, the minimum supported level is Java SDK/JRE/JDK 7.0.17. For the JDK from Oracle on Mac OS X, the minimum supported level is Java SDK/JRE/JDK 7.0 Update 15.

### 8.5.5.5 Java SE 8 runtime environment

For the Java SDK from IBM, the minimum supported level is IBM SDK, Java Technology Edition, Version 8. For the JDK from Oracle, the minimum supported level is Java 8 update 25.

### Distributed operating systems

On distributed platforms, 32-bit or 64-bit Java is supported.

### Distributed operating systems

For Windows and Linux systems, you can use either the JDK from Oracle or the JDK from IBM. If you are developing applications on Windows or Linux, and you plan to deploy those applications to a server running on WebSphere Application Server classic, you should use the JDK from IBM. For HP systems and Mac OS, use the JDK from Oracle.

**Note:** **IBM i** To obtain the minimum supported Java level for IBM iSeries, install IBM J2SE 6.0 32-bit JVM (5761-JV1 option 11 **8.5.5.2** or 5770-JV1 option 11) or IBM SE 6.0 64 bit (5761-JV1 option 12 **8.5.5.2** or 5770-JV1 option 12), and also install Java PTF group SF99572 level 8 (or later) for IBM i 7.1.

## The installation directory name and path cannot include non-ASCII characters

Recent JVMs do not fully support use of non-ASCII characters with the **-jar** and **-javaagent** commands. You should use only ASCII characters in your installation directory names and paths.

## Changing the JDBC data source at run time might result in JPA failures

If the database dictionary type is not specified through properties, it is detected and calculated by OpenJPA when the first entity manager is created and the database connection is made. This database dictionary type is used for all entity managers that are subsequently created. If the JDBC data source is changed while an application is running, the entity manager factory does not detect this change and continues to use the old dictionary for operations against the new data source. This can result in failures if the database is changed to a different vendor.

When you change a database to a different vendor, restart the application.

## Modifying the dataSource, jdbcDriver, connectionManager, and JDBC vendor properties at run time might result in JPA failures

If you update the configuration of dataSource, jdbcDriver, connectionManager or any of the JDBC vendor properties lists (for example, properties.db2.jcc or properties.oracle) while the server is running, you might see J2CA8040E failures. These failures state that multiple dataSource elements cannot be associated with a single connectionManager. These failures are generated even if your configuration associates only one connectionManager with the dataSource element.

When you make updates to the configuration of any of these JDBC resources, restart the server.

## **An application that relies on a result being returned by `getRealPath` must be deployed as an expanded application, not as a WAR file**

The Java EE specification states that the `getRealPath()` method returns a null value if the content is being made available from a web archive (WAR) file. When you deploy a WAR file to Liberty, Liberty does not automatically extract the archive file into a directory structure. Therefore the application might fail to start. If your application relies on a result being returned by `getRealPath()`, you must deploy the application as an expanded web application, not as a WAR file. For example, you can manually extract the WAR file and copy the expanded application to the `dropins` directory.

## **WebSphere Application Server traditional scripts do not work with Liberty**

You cannot use any of the scripts under the `bin` directory of the WebSphere Application Server traditional to administer Liberty.

## **Fileset restrictions**

The following restriction applies to Filesets:

- Filesets do not recursively explore subdirectories of the base directory. For example, the following instructions are not supported:

```
<fileset id="testFileset" dir="\temp" includes="**\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="*\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\b\a.jar"/>
```

## **Overriding classes from the Java SDK**

Some Java EE 6 technologies supported by Liberty require newer versions of APIs that are provided by Java SE 6. JAX-WS, JAXB and the `javax.annotation.Resource` annotation are all examples where a Java 6 SDK contains older classes than those required by Java EE 6. When compiling your application code against these APIs using Java SDK version 6, you need to use classes that are provided by Liberty rather than the Java SDK. You must take one of the following actions:

- If you are using `javac` to build from the command line, compile your code by using the `javac -endorseddirs` option and the JAR files in the `${wlp.install.dir}/dev/specs` directory.
- If you are using Apache Ant to build, compile your code by using the `<compilerarg>` child element of the `javac` task and the JAR files in the `${wlp.install.dir}/dev/specs` directory. In the build script, specify the `-endorseddirs` option and the `${wlp.install.dir}/dev/specs` directory as separate `<compilerarg>` elements. For example:

```
<javac srcdir="src" destdir="classes"/>
 <compilerarg value="-endorseddirs"/>
 <compilerarg value="${wlp.install.dir}/dev/specs"/>
</javac>
```

- If you are using Apache Maven to build, compile your code by using the `endorseddirs` element of the Maven compiler plug-in and the JAR files in the `${wlp.install.dir}/dev/specs` directory. For example:

```
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <configuration>
 <source>1.6</source>
 <target>1.6</target>
 <compilerArguments>
```

```
 <endorseddirs>${wlp.install.dir}/dev/specs</endorseddirs>
 </compilerArguments>
</configuration>
</plugin>
```

#### Windows

## When you unpublish a shared library, it cannot be deleted until the server is stopped

When you unpublish a shared library from a server, the library JAR file is not immediately released from the server. Therefore the operating system does not know that the file is no longer in use, and does not let you delete the file. When you next stop the server, the library JAR file is released and you can delete the file.

## java:global lookups restrictions

Resources defined in applications with `java:global` lookups can only be used to access names declared by applications deployed in the current server.

## appSecurity-2.0 feature restrictions

For the `appSecurity-2.0` feature, the following restrictions apply:

- For EJB applications, the **run-as-mode** of `SYSTEM_IDENTITY` is not supported in the extension settings of the `ibm-ejb-jar-ext.xml` file.
- The `getCallerIdentity` API is not supported for Singleton session beans.
- Role names can be referenced by the `HttpServletRequest.isUserInRole` and `EJBContext.isCallerInRole` APIs or by elements in the deployment descriptor without first declaring the role names using the `@DeclareRoles` annotation or the `<security-role/>` element in the deployment descriptor. However, roles must be declared before being used in WebSphere Application Server traditional.

## Applications not starting in an embedded Liberty server

Ensure that the Java process that starts the embedded Liberty server was started with the `-javaagent` JVM argument that pointed to the `libertyInstallDir/bin/tools/ws-javaagent.jar`. If the `-javaagent` JVM argument is not used the server runtime starts, but applications fail to start with no obvious exceptions.

#### 8.5.5.6

## WebSphere MQ resource adapter and generic JCA support related restrictions

The WebSphere® MQ resource adapter can be used within the WebSphere Application Server Liberty by using either the `wmqJmsClient-1.1` or `wmqJmsClient-2.0` feature or by using generic JCA support.

You can use the WebSphere MQ resource adapter version 7.5 with Liberty version 8.5.5 and later. If you want to use WebSphere MQ resource adapter version 8.0, which is based on JMS 2.0 resource adapter, you must ensure that you are using the latest Liberty version that is compatible with the JMS 2.0 resource adapter.

### Notes:

- With Liberty version 8.5.5.2, the `wmqJmsClient-1.1` feature must be used with a IBM MQ resource adapter version 7.5.0.5 or later.
- With Liberty version 8.5.5.6, the `wmqJmsClient-2.0` feature must be used with a IBM MQ resource adapter version 8.0.0.3 or later

To know more about the version compatibility information between WebSphere MQ resource adapter and Liberty, see the Reference to obtain the WebSphere MQ resource adapter.

If you are using generic JCA support, the following restrictions apply:

- To run the IBM® WebSphere MQ resource adapter on z/OS, you must use the `wmqJmsClient-1.1` or `wmqJmsClient-2.0` feature.
- Tracing and logging are not integrated within the Liberty trace system using generic JCA. Trace is written to a separate file, and it must be enabled by setting the system properties. The procedure to enable tracing is the same as configuring the WebSphere MQ classes for JMS trace facility for a Java Standard Environment. See Java Standard Environment Trace stanza.
- The IBM MQ classes for Java are not supported in Liberty. They must not be used with either the IBM MQ Liberty messaging feature or with the generic JCA support. For more information, see Using WebSphere MQ Java Interfaces in J2EE/JEE Environments.

## Versioning is not possible for applications in the “dropins” directory

For applications in the “dropins” directory, the file name and file extension are used by the application monitor to determine the type of application, and to generate the application id and application name. It is therefore not possible to specify the version number for the application by using the file name or file extension. In a production environment, you are not recommended to use the “dropins” directory.

## Shared session applications must store session objects in shared libraries

When you are using the `shared-session-context` application extension, or `<shared-session-context value="true"/>` in `ibm-application-ext.xml`, all objects that are stored in the session must be available in the shared libraries that are associated with the application so that the session can be invalidated.

## Configuring session persistence

There is only one `server.xml` file for each server, not a `server.xml` file for each EAR or WAR file. You set session persistence in a database by adding:

```
<httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS" ... />
```

In Liberty, this setting for the database applies to all EAR and WAR files. It is not possible to set up some databases with session persistence and others without.

## Ported locally transacted JMS sessions do not work in Liberty

In WebSphere Application Server traditional, you can develop applications to take advantage of locally transacted JMS sessions. When you port these applications to Liberty, these applications behave differently or do not work at all.

Even though WebSphere Application Server traditional allows locally transacted JMS sessions, porting a WebSphere Application Server traditional locally transacted JMS session to Liberty is not allowed.

## Admin Center feature restrictions

For the `adminCenter-1.0` feature, the following restriction applies:

- **8.5.5.2** Using an IBM Java virtual machine (JVM) available with a WebSphere Application Server traditional product, such as Network Deployment, can cause WebSphere Liberty Administrative Center (“Admin Center”) to not display in a browser. By default, the IBM JVM available with the WebSphere Application Server traditional product points to security classes that are available only with the WebSphere Application Server traditional product, and not to security classes needed by Admin Center, which requires Secure Sockets Layer (SSL). Use a JVM that supports Liberty products and SSL.

**8.5.5.4** You can get a JVM that supports Liberty products and SSL from Installation Manager offerings or developerWorks:

- Using Installation Manager, select the Liberty product first and then select WebSphere SDK for Liberty. Use Installation Manager to install the Liberty product and software development kit (SDK). The WebSphere SDK for Liberty includes the needed support for Liberty products and SSL and offers a Java client, JConsole.
- Go to <http://www.ibm.com/developerworks/java/jdk/index.html> on the developerWorks website and download an IBM Java development kit (JDK) for your operating system. The developerWorks website does not have a JVM for all operating systems. For example, you have to get the JDK from Eclipse for Windows operating systems.
- **8.5.5.4** From the Admin Center Deploy tool, you cannot use the **Use the connection method and credentials configured for each target host** option of **Remote Management Credentials** to deploy Liberty 8.5.5.3 or earlier server package to a registered host. The server package must support Liberty 8.5.5.4 or later.
- **8.5.5.5** The CPU Usage chart of the Admin Center Monitor view shows either **0%** or **null%** CPU usage for JVMs that do not supply process CPU statistics. For information about the chart, see “Monitoring metrics in Admin Center” on page 1087.

## Bean validation feature restrictions

For the beanvalidation-1.0 feature, the following restriction applies:

- There is no support for bean validation inside OSGi applications.

**8.5.5.6** For the beanValidation-1.1 feature, the following restrictions apply:

- There is no support for bean validation inside OSGi applications.
- Applications that provide a custom ConstraintValidatorFactory implementation in a validation.xml file with the beanValidation-1.0 feature do not compile against the Bean Validation 1.1 API.
- If a validation.xml file is not located in the module it is associated with, then there can be only one validation.xml file and the `com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp` property must be set to `false` in either of the following files:
  - `jvm.options`
    - `-Dcom.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false`
  - `bootstrap.properties`
    - `com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false`

## Dynamic cache feature restrictions

The following dynamic cache features are not available or have limited availability:

- Cache replication is not supported.
- Only high performance disk caching mode is supported with random and size based eviction techniques.
- There is no support for Web Services client and server side caching as well as portlet cache in the `cachespec.xml` file.
- There is no support for servlet caching of `SingleThreadModel` servlets.
- Defining cache configuration by using properties files is not supported for JAR files that contain only Enterprise JavaBeans (EJBs).
- Limiting a heap cache size works only for 32-bit Java virtual machines (JVM).

## ejbLite-3.1 feature restrictions

For the ejbLite-3.1 feature, the following restrictions apply:

- EJB modules prior to version 3.0 are not supported. This restriction also means that bindings and extensions using the .xmi file format rather than the .xml file format are not supported.
- Session beans are not bound to the ejblocal namespace, which means JNDI lookups and ejb-ref binding names must use java:global, java:app, or java:module names. The simple-binding-name and interface binding-name elements are ignored in ibm-ejb-jar-bnd.xml files.
- The stateful bean passivation directory is not configurable. Files are passivated to the server work area.

8.5.5.6

### **jpa-2.1 feature restrictions**

For the jpa-2.1 feature, JPA entity exchange over CORBA/RMI-IIOP requires that both participants in the communication must enable identical JPA feature levels.

### **jsp-2.2 feature restrictions**

For the jsp-2.2 feature, the following restriction applies:

- There is no support for the useInMemory configuration option to only store the translated JSP file in memory.

8.5.5.9

### **logstashCollector-1.0 feature restrictions**

The following limitations apply to the logstashCollector-1.0 feature:

- Data Loss – Some events that are generated in Liberty might not be forwarded to Logstash as expected. Data loss might occur under the following scenarios:
  1. Starting the Liberty server before the Logstash server is started. It is recommended that you start the Logstash server before starting the Liberty server.
  2. Heavy load conditions. Events might be dropped in cases where events are created in Liberty faster than they can be processed by Liberty's event pipeline, Logstash, and any other downstream consumers. Liberty uses buffers to avoid data loss when event creation is briefly faster than event consumption.
- The logstashCollector-1.0 feature is tested and is compatible with Logstash V2.x.

### **monitor-1.0 feature restriction**

For the monitor-1.0 feature, the following restriction applies:

- When the feature is removed from the server.xml file, you must restart the server to make the JAX-WS applications work.

8.5.5.6

### **requestTiming-1.0 feature restrictions**

For the requestTiming-1.0 feature, the following restriction applies:

- The requestTiming-1.0 feature, when activated, has been shown to have an 4% impact on maximum possible application throughput when measured with the DayTrader application. While the impact on your application might be more or less than that, you should be aware that some performance degradation might be noticeable.

8.5.5.6

### **restConnector-1.0 feature restriction**

For the restConnector-1.0 feature, the following restriction applies:

- Users of the `restConnector-1.0` feature or any feature that includes `restConnector-1.0`, such as `collectiveMember-1.0` and `collectiveController-1.0`, who want to run applications containing a custom JAXRS 2.0 runtime must add the `jaxrs-2.0` feature to that server.

8.5.5.8

### **scim-1.0 feature restrictions**

The following restrictions apply for the `scim-1.0` feature:

- The `members` attributes are not retrieved while searching for groups.
- The `groups` attributes of users are not retrieved while searching for users.
- The Canonical type of `direct/indirect` cannot be set for groups attribute of users.
- Only one `email` attribute of user of Canonical type, `work`, can be defined.
- Only one `ims` attribute of user of Canonical type, `work`, can be defined.
- Extended schema attributes of SCIM such as `entitlements`, `roles` and `x509Certificates` cannot be set or returned.
- The `userName` attribute cannot be used with some other attributes in a filter.
- For users in Basic and SAF registries, only `userName`, `displayName`, `id`, `schema`, `meta.location` and `groups` can be set. The `userName` and `displayName` will have the same value.
- List/query with Basic and SAF registries does not work the same as the `ldapRegistry` registry.
- Operators like `pr`, `gt`, `ge`, `lt`, `le`, and, `or`, and `()` will not work with Basic and SAF registries. Also, only one operator must be used in the filter for Basic and SAF registries.
- Basic and SAF are read only registries.
- While creating user, `groups` attribute cannot be set.

### **wmqJmsClient-1.1 feature restrictions**

For the `wmqJmsClient-1.1` feature, the following restrictions apply:

- You must manually set the `PATH` variable in the Windows environment variables to point to the IBM MQ installation bin directory. You must set this path variable when the application uses the `BINDING` connection mode.
- The IBM MQ classes for Java (generally called the Base Java) are not included in the `wmqJmsClient-1.1` feature. This is included in the Resource Adapter for other application servers but is not recommended for the Base Java APIs in the Java Enterprise Edition environments. For more information, see *Using IBM MQ Java Interfaces in J2EE/JEE Environments*.
- The `BINDINGS_THEN_CLIENT` transport type of IBM MQ resource adapter is not supported for the `wmqJmsClient-1.1` feature.
- The Advanced Messaging Security (AMS) feature is not included for the `wmqJmsClient-1.1` feature.

8.5.5.6

### **wmqJmsClient-2.0 feature restrictions**

For the `wmqJmsClient-2.0` feature, the following restrictions apply:

- You must manually set the `PATH` variable in the Windows environment variables to point to the IBM MQ installation bin directory. You must set this path variable when the application uses the `BINDING` connection mode.
- The IBM MQ classes for Java (generally called the Base Java) are not included in the `wmqJmsClient-2.0` feature. This is included in the Resource Adapter for other application servers but is not recommended for the Base Java APIs in the Java Enterprise Edition environments. For more information, see *Using IBM MQ Java Interfaces in J2EE/JEE Environments*.
- The `BINDINGS_THEN_CLIENT` transport type of IBM MQ resource adapter is not supported for the `wmqJmsClient-2.0` feature.

## concurrent-1.0 feature restrictions

8.5.5.4 For the concurrent-1.0 feature, the following restrictions apply:

For the thread context of type `securityContext`, any custom information in the subject that was not added by using a JAAS login module will not be propagated. For example, if the submitter's subject contains a custom Principal that was added by a TAI, the propagated subject will not contain this custom Principal.

8.5.5.6

## jacc-1.5 feature restrictions

For the jacc-1.5 feature, the following configurations are ignored:

- Authorization information (the users and groups attributes of the authorizations attribute) in an `ibm-application-bnd.xml` file or an `ibm-application-bnd.xmi` file of the application's ear file.
- Authorization information (the user, group and special-subject attributes of the security-role attribute in the `application-bnd` element) in the `server.xml` file.

---

## Developer Tools known issues and restrictions

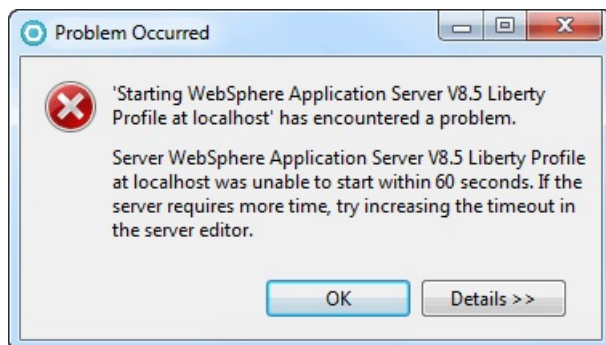
### Distributed operating systems

Several known issues and restrictions apply when you are working with WebSphere Application Server Developer Tools for Eclipse.

See also “Runtime environment known issues and restrictions” on page 1480.

### Unable to interact with the Liberty server after you modify the console log level settings

There is a known limitation when the console log level is set to WARNING, ERROR, or OFF. The workbench has problems when it interacts with the Liberty server, such as cannot start, stop, or publish to the server. For example, the workbench is unable to start the Liberty server and the following timeout error message displays:



The console log level (`consoleLogLevel`) is an attribute of the logging configuration element in the server configuration (`server.xml`) file with the following range options: INFO, AUDIT, WARNING, ERROR, and OFF. AUDIT is the default value for the console log level settings. For more details, search for the `consoleLogLevel` attribute in the Configuration elements in the `server.xml` topic.

To work around this known limitation, specify INFO or use the default AUDIT setting for the console log level:



1. In the Servers view, expand your Liberty server.
2. Right-click the **Server Configuration**[server.xml] node and select **Open**.
3. In the Server Configuration editor and under the **Configuration Structure** section, expand **Server Configuration** node. The next step depends if the **Logging** element is available:
  - If the **Logging** element is available, select it and under the **Logging** section of the server configuration editor, use the drop-down menu for the **Console log level** field, and select either the **AUDIT** or **INFO** option. Type **Ctrl + s** to save your changes in the editor.
  - If the **Logging** element is not available, the workbench is already using the default **AUDIT** setting. As a result, you might be experiencing a different problem that is causing interaction failures between the workbench and the Liberty server.

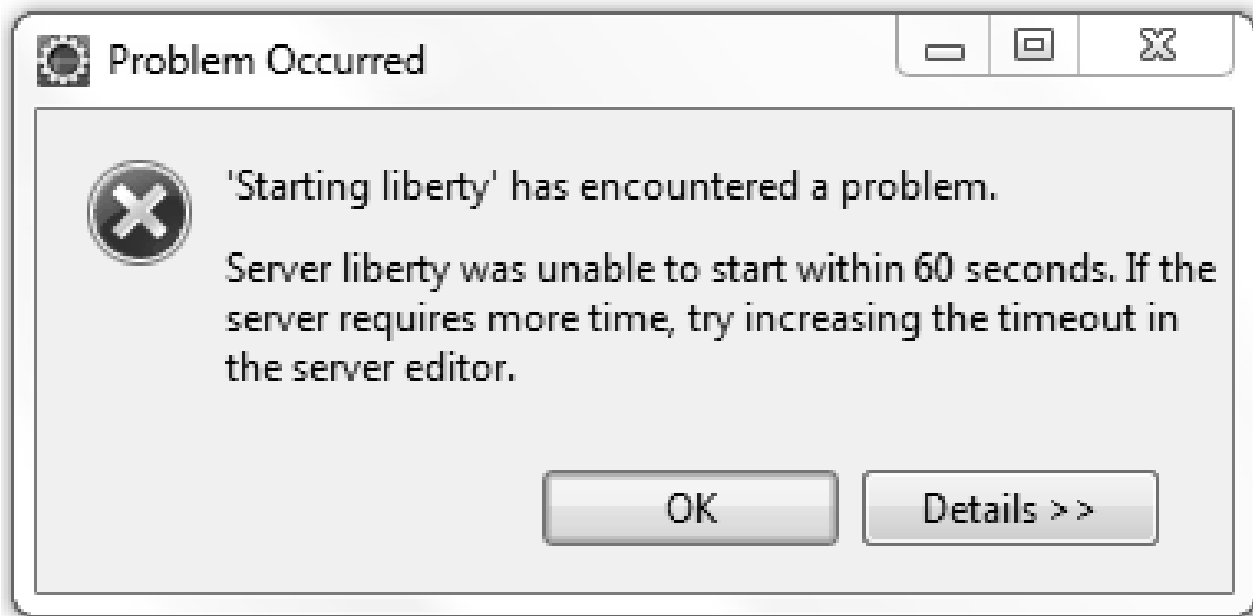
## Copying and pasting servers might cause the publishing state to become out of synchronization

Try to avoid copying and pasting servers because they point to the same configuration file. Copying and pasting servers might cause the publishing state to become out of synchronization. For example, when you remove an application from one server, the application still appears to be deployed to the other server even though it is not. Server state will not synchronize up again until the next publish operation.

8.5.5.4

### Unable to recognize the start of the server when the hideMessage attribute is used to suppress the messages

You can configure the `<hideMessage>` attribute in the **Logging** element of **Server Configuration** [server.xml] to suppress messages. If you configure to hide the server start message, for example `<logging hideMessage="CWWKF00111"/>`, then the tool fails to recognize the state of the server when it is started. In such a situation, the state of the server in the **Server** view remains as *starting* until timeout and finally displays the following message:



8.5.5.4

## Remote servers cannot run projects that were created with a newer version of Java

If you compile a project with a higher version of Java than your remote server is running, you can receive the following error messages:

```
Error 404: javax.servlet.UnavailableException:
SRVE0202E: Servlet [s1]: s1 was found, but is corrupted:
SRVE0227E: Check that the class resides in the proper package directory.
SRVE0228E: Check that the classname has been defined in the server using the proper case and fully qualified package.
SRVE0229E: Check that the class was transferred to the filesystem using a binary transfer mode.
SRVE0230E: Check that the class was compiled using the proper case (as defined in the class definition).
SRVE0231E: Check that the class file was not renamed after it was compiled.
```

If you create a project in a package with a higher version of Java than your remote server is running, you can receive the following error messages:

```
Error 404: java.io.FileNotFoundException: SRVE0190E: File not found: /s2
Console output: [ERROR] SRVE0266E: Error occurred while initializing servlets: java.lang.UnsupportedClassVersionError:
at java.lang.ClassLoader.defineClassImpl(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:383)
at com.ibm.ws.classloading.internal.AppClassLoader.definePackageAndClass(AppClassLoader.java:318)
at [internal classes]
```

To avoid these errors, compile your project with the same version of Java that your remote server is running, or with an earlier version. You can also create your project in a package with the same version of Java that your remote server is running, or with an earlier one. To find the Java version of the remote server, check the messages.log file. For more information about viewing the messages.log file, see “Viewing trace and message log files by using developer tools” on page 1465.

8.5.5.10

## Some utilities are unavailable when you use a remote Liberty server or a Liberty server in a Docker container

WebSphere Developer Tools do not support some of the utilities when you use a remote Liberty server or Liberty server in a Docker container.

The tools do not support the following utilities when you use a remote Liberty server:

- **Generate Web Server Plug-in...**
- **Add Config Snippets...**

The tools do not support the following utilities when you use a Liberty server in a Docker container:

- **Generate Web Server Plug-in...**
- **Package Server...**
- **Create Collective...**
- **Add Config Snippets...**

---

## Messages

When you use Liberty, you might encounter system messages. Each message has a unique message identifier, and includes an explanation of the problem, and details of any action that you can take to resolve the problem.

Liberty system messages are logged from various sources, including application server components and applications. For Liberty, the message identifier is 10 characters in length and has the following form:

```
CWXXX9999X
```

where:

**CWXXX**

Is a five-character alphabetic message prefix that identifies the Liberty component.

**9999** Is a four-character numeric identifier used to identify the specific message for that component.

**X** Is an optional alphabetic indicator that identifies the message type: I=Informational, W=Warning, E=Error.

The Troubleshooter reference: Messages topic contains information about all WebSphere Application Server messages, indexed by message prefix. For each message there is an explanation of the problem, and details of any action that you can take to resolve the problem.

*Table 108. Alphabetic message prefixes for Liberty messages*

Liberty message prefix	Range	Liberty component
CWIMK	0001-0500	LDAP registry federation (configuration messages)
CWIML	0001-5000	LDAP registry federation (runtime messages)
CWLIB	0001-0100	Transactions
	0101-0200	z/OS Transaction Extensions
CWWKB		<i>z/OS Native integration</i>
	0001-0050	z/OS Command Processing
	0051-0100	z/OS Native code only
	0101-0150	z/OS Core
	0151-0200	z/OS WLM services
	0201-0250	z/OS Native code non-console
CWWKC	0000-0250	Annotation scanning
CWWKE		<i>Core kernel and kernel services</i>
	0001-0099	Boot/Launcher
	0100-0199	Service utilities
	0200-0299	Location service
	0300-0399	File installation service
	0400-0499	FileMonitor service
	0500-0599	Extensible Class Scanner
CWWKF		Feature manager
CWWKG		Configuration manager
CWWKJ		Light touch administration
CWWKL	0001-0100	Class loading service

Table 108. Alphabetic message prefixes for Liberty messages (continued)

Liberty message prefix	Range	Liberty component
CWWKM	0001-0050	Artifact API messages (container factory)
	0051-0100	Overlay Messages (all implementations)
	0101-0150	Artifact API Zip implementation
	0151-0200	Artifact API File implementation
	0401-0450	Adaptable API Implementation messages (adaptable module factory / adapter service)
	1001-1100	Loose archive API (used by the Eclipse-based tools option that runs an application directly from the Eclipse workspace(The Eclipse platform supports a "virtual" application archive, where a set of files that appear to be in the same archive file are actually spread out across the Eclipse workspace. Liberty calls this a "loose archive", and the tools option that uses the loose archive API is called <b>Run this application directly from the workspace</b> )).
	2000-2999	Ant and Maven plug-in
CWWKN	0001-0100	JNDI default namespace
CWWKO		<i>CFW components</i>
	0000-0199	CFW
	0200-0399	TCP
	0400-0599	UDP
	0600-0799	bytebuffer
	0800-0899	SSL channel
	0900-0999	SSL

Table 108. Alphabetic message prefixes for Liberty messages (continued)

Liberty message prefix	Range	Liberty component	
CWWKS		Security	
	0000 series	Security: General messages	
		0000-0099	Security
		0900-0999	Security quickStart
	1000 series	Security: Authentication services	
		1000-1099	Authentication
		1100-1199	JAAS authentication
		1200-1299	TAI authentication
	2000 series	Security: Authorization services	
		2000-2099	Authorization
		2100-2199	Built-in authorization
		2900-2999	SAF authorization
	3000 series	Security: Registry services	
		3000-3099	Registry
		3100-3199	Basic registry
		3200-3299	LDAP registry
		3300-3399	FileBased (VMM) registry
		3800-3899	Custom registry
		3900-3999	SAF registry
	4000 series	Security: Token services	
		4000-4099	Token services
		4100-4199	LTPA
		4200-4299	Kerberos
		4300-4399	SPNEGO
	9000 series	Security: Collaborators	
		9100-9199	Web Collaborator (common code)
		9200-9299	Web Application Collaborator
	9300-9399	Web Administration Collaborator	
CWWKT		HTTP transport/dispatcher	
CWWKX	0000 series	JMX	
		0001-0100	JMX Security
		0101-0200	JMX REST Connector
		0201-0300	JMX REST Client
	8.5.5.3 1000 series		Administrative Center
		8.5.5.3 1000-1899	Admin Center

Table 108. Alphabetic message prefixes for Liberty messages (continued)

Liberty message prefix	Range	Liberty component
CWWKZ		<i>Applications</i>
	0001-0100	Application manager
	0101-0200	WARs
	0201-0300	WABs
	0301-0400	EBAAs

## Troubleshooting OSGi applications by using developer tools

8.5.5.9

You can configure the `osgiAppConsole-1.0` feature with WebSphere Developer Tools and then use it to determine and analyze the OSGi applications-related issues.

### Before you begin

- Install WebSphere Developer Tools for Eclipse and WebSphere Application Server Liberty. You can install the tools by using downloaded installation files. For more information about installing WebSphere Developer Tools and Liberty, see “Installing Liberty developer tools and (optionally) Liberty” on page 835.
- Deploy an OSGi application to a running Liberty server.
- The `osgiAppConsole-1.0` feature is available as part of the downloaded compressed files. If the feature is not already installed, then install the feature from the Liberty Repository by using the following command:

```
bin\installUtility install osgiAppConsole-1.0
```

### Procedure

1. Configure the `osgiAppConsole-1.0` feature in the `server.xml` file.

```
<featureManager>
 <feature>osgiAppConsole-1.0</feature>
</featureManager>
```
2. Add the security configurations.

The following example shows a sample security configuration with an `admin` role that uses basic authorization and HTTPS security.

```
<httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint"/>
<keyStore id="defaultKeyStore" password="Liberty"/>
<quickStartSecurity userName="admin" userPassword="password"/>
```
3. Start the OSGi Application Console or the OSGi Shared Bundle Console by completing one of the following tasks.
  - Start the OSGi Application Console:
    - a. In the Servers view, right-click the OSGi module.
    - b. Select **Launch OSGi Application Console**.

The tools open a browser that displays the OSGi Application Console page.
  - Start the OSGi Shared Bundle Console:
    - a. Right-click your server.
    - b. Select **Utilities > Launch OSGi Shared Bundle Console**.

The tools open a browser that displays the OSGi Shared Bundle Console page.

---

## Troubleshooting Session Initiation Protocol (SIP) on Liberty

Troubleshoot SIP by using the available log and trace facilities. You can use the default log and trace facility, or you can enable binary logging for high-performance, extensible logging and tracing.

### Troubleshooting the SIP container session repository on Liberty

8.5.5.7

When you troubleshoot the SIP container session repository, you might need the SIP session details to dump to a specified trace file.

#### About this task

You can use the SIP session memory dump utility to help debug problems that are related to SIP container sessions. The SIP container provides the `SipContainerMBean` method to perform several serviceability type operations on the SIP container, including the initiation of a server quiesce through the command line. This task describes how you can use the `SipContainerMBean` method to dump SIP application session and SIP session information that is contained in the in-memory session repository for SIP containers. By configuring the `SipContainerMBean` method to use various trace methods, you can specify the SIP session details to dump to the specified trace file.

When the session dump methods are started, the requested information about the sessions prints by default into the `console.log` file. You can also send the information to a predefined source specified on the `setDumpMethod` method.

You can run the dumping utility in two modes, *succinct* and *verbose*. When you use the succinct session dump methods, only the session IDs are printed for every dump method execution. If you want to use the verbose session dump methods, the following actions occur:

- Transaction user details, along with the SIP session details, if they exist, print for every dump method execution.
- The only attributes that dump to the trace file are those attributes that the JSR 289 specification permits for exposure.
- The verbose methods print the following information in the trace file: `appName`, `callID`, `dialog state`, `creation time`, `attribute names`.

The trace printouts occur per SIP application; therefore, the sorting of all the SIP session data structures occurs before printing. The `SipContainerMBean` dump facility runs in a low-priority thread so that the tracing does not affect the call processing latency of the overall system for a production server.

The dump distinguishes between a transaction user that has a SIP session versus a transaction user that has no `SipSession` object. Also included in the dump, in a delineated fashion, are SIP sessions that no longer exist, that are no longer valid, or that exist at the time of the trace snapshot.

On Liberty, you can invoke the `SipContainerMBean` dumping methods in two ways:

- By running the **server dump** command
- By implementing a Java Management Extensions (JMX) client that establishes a connection to the JMX connector to invoke the methods

The following succinct `SipContainerMBean` methods are used to dump SIP session IDs.

Table 109. Succinct `SipContainerMBean` methods used to dump SIP session information.

Method	Description
<code>dumpAllSASIDs()</code>	Prints a number of all SIP application sessions and the SIP application session IDs.

Table 109. Succinct SipContainerMBean methods used to dump SIP session information (continued).

Method	Description
dumpAllTUSipSessionIds()	Prints a number of transaction users and the SIP session IDs within the transaction user (TU), if one exists.

The following verbose SipContainerMBean methods are used to dump SIP session details.

Table 110. Verbose SipContainerMBean methods used to dump SIP session information.

Method	Description
dumpAllSASDetails()	Prints a number of all SIP application sessions and the SIP application session ID details.
dumpAllTUSipSessionDetails()	Prints a number of transaction users and details of the SIP session IDs within the transaction user (TU), if one exists.
dumpSASDetails(String sasId)	Prints the details of the SIP application session that is specified by the sasId parameter.
dumpSipSessionDetails(String sessionId)	Prints the details of the SIP session that is specified by the sessionId parameter.

**Note:** Use the following information to help parse print output:

- For all print output, the first line provides an application name and a number of records.
- The delimiter between the output is a TAB.
- The delimiter between session attributes is a ; (semi-colon).

## Procedure

- Invoke the SipContainerMBean methods with the **server dump** command. For more information about the **server dump** command, see “Generating a Liberty server dump from the command line” on page 960.
  1. In the server.xml file, specify the dumping utility mode on a sipIntrospect element.
    - For succinct mode, specify the following code:
 

```
<sipContainer>
 <sipIntrospect method="SUCCINCT"/>
</sipContainer>
```
    - For verbose mode, specify the following code:
 

```
<sipContainer>
 <sipIntrospect method="VERBOSE"/>
</sipContainer>
```
  2. Open a command line and change to the wlp/bin directory.
  3. To generate the SIP sessions and SIP application sessions dump, run the following command, where *server\_name* is the Liberty server:
 

```
server dump server_name
```

 If you do not specify a server name, defaultServer is used.  
 You can optionally specify the name of the server dump package file on the **archive** parameter:
 

```
server dump server_name --archive=package_file_name.dump.zip
```

 After you run the command, the server dump package file is generated and contains the following files:
    - SipContainerIntrospector.txt, which includes the requested level of trace information about the SIP application sessions and SIP sessions
    - Other *artifact*Introspector.txt files, which provide the server status information



SipContainerIntrospector.txt includes the following information, where dump.ids.test.app1 is the application name, 2 is the number of sessions, and the local.##### lines are the SAS\_ids:

The description of this introspector:  
SIP state details

Succinct dumping

```
dump.ids.test.app1 2
local.1347524282775_8
local.1347524282775_7
```

--- End of Dump ---

- Invoke a specific dumping utility method through JMX connectors. For more information, see “Connecting to Liberty by using JMX” on page 1021 and “Developing a JMX Java client for Liberty” on page 1024.

The following Java code example invokes the dumpAllSASIds method:

```
System.setProperty("javax.net.ssl.trustStore", "..\\wlp\\usr\\servers\\SIPServer\\resources\\security\\key.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "Liberty");

//If the type of the trustStore is not jks, which is default,
//set the type by using the following line.
System.setProperty("javax.net.ssl.trustStoreType", "jks");

try {
 HashMap<String, Object> environment = new HashMap<String, Object>();
 environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
 environment.put("com.ibm.ws.jmx.connector.client.disableURLHostnameVerification", Boolean.TRUE);
 environment.put(JMXConnector.CREDENTIALS, new String[] { "theUser", "thePassword" });
 environment.put(ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION, true);

 JMXServiceURL url = new JMXServiceURL("service:jmx:rest://localhost:9443/IBMJMXConnectorREST");
 JMXConnector connector = JMXConnectorFactory.newJMXConnector(url, environment);
 connector.connect();
 MBeanServerConnection connection = connector.getMBeanServerConnection();

 // test dumping utility
 connection.invoke(new ObjectName("WebSphere:name=com.ibm.ws.sip.container.SipContainerMBean"), "dumpAllSASIds",
```

Succinct SipContainerMBean methods have the following print formats.

Table 111. Succinct SipContainerMBean method print formats.

Method	Print format
dumpAllSASIds()	Each line represents a SIP application session ID, in the format [SAS_ID].  For example: local.1347524282775_8
dumpAllTUSipSessionIds()	Each line represents the transaction ID, whether the transaction has a SIP session, and the session ID if it exists, in the format [TU_ID] [hasSIPSession] [SipSessionId]  For example: local.1349965420866_1_0 true local.1349965420866_1_0 1

Verbose SipContainerMBean methods have the following print formats.

Table 112. Verbose SipContainerMBean method print formats.

Method	Description
dumpAllSASDetails()	<p>Each line represents the SIP application session ID, the creation time, and the SIP session attributes, in the format [SAS_ID] [CreationTime] [attributes]</p> <p>For example:                      local.1348147884986_2 Sep 20,2012 16:31 DumpSasDetailsAttr;</p>
dumpAllTUSipSessionDetails()	<p>Each line represents the transaction ID, whether the transaction has a SIP session, and the session details if it exists, in the format [TU_ID] [hasSIPSession] [SipSessionId] [Call-Id] [DialogState] [hasOutgoingTransaction] [initialMethod] [SAS_ID] [CreationTime] [attributes]</p> <p>For example:                      local.1349965420866_1_0 true local.1349965420866_1_0_1 8-8548@9.148.57.128 2 false INVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAttr2;</p>
dumpSASDetails(String sasId)	<p>Only one line is printed, in the format [TU_ID] [hasSIPSession] [SipSessionId]</p> <p>For example:                      local.1349965420866_1_0 true local.1349965420866_1_0_1</p> <p>If the requested session does not exist, the following error message is printed:                      ERROR: Requested session &lt;local.1349965420866_1_0_1&gt; does not exist.</p>
dumpSipSessionDetails(String sessionId)	<p>Only one line is printed, in the format [SipSessionId] [Call-Id] [DialogState] [hasOutgoingTransaction] [initialMethod] [SAS_ID] [CreationTime] [attributes]</p> <p>For example:                      local.1349965420866_1_0_1 8-8548@9.148.57.128 2 false INVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAttr2;</p> <p>If the requested session does not exist, the following error message is printed:                      ERROR: Requested session &lt;local.1349965420866_1_0_1&gt; does not exist.</p>

## Tracing a Session Initiation Protocol (SIP) container on Liberty

8.5.5.7

You can trace a Session Initiation Protocol (SIP) container, starting either immediately or after the next server start. This tracing writes a record of SIP events to a log file.

### Procedure

In the `server.xml` file, add a logging element and specify tracing for the SIP container by setting `traceSpecification="*=info:com.ibm.ws.sip=all"`.

```
<logging logDirectory="${server.config.dir}/logs" traceFileName="trace.log" traceSpecification="*=info:com.ibm.ws.sip=all"/>
```

### Results

SIP-level tracing messages appear in `serverName/logs/trace.log`, where `serverName` is the name of the specific instance of the application server that is running the SIP container to be traced. These messages

include application load events as well as SIP request and response parsing and SIP servlet invocation.

## Session Initiation Protocol (SIP) binary log and trace extensions on Liberty

8.5.5.7

Binary logging provides a way for developers to add extension fields to log and trace records, and a corresponding way for you to filter log and trace records by extension value.

Log and trace records contain fields for information such as the time the record was created and the content of the message that is logged. These fields are core fields that are present in every log and trace record. In contrast, extension fields are fields which application developers can add to log and trace records, which you can use as filter criteria when you search for specific log and trace content. These log and trace extensions are visible in the binary log when you configure the text output format to use the advanced format, or they are visible when you use the `binaryLog` command in the advanced format.

### Administrators

The application server automatically creates a number of extensions that you can use to filter log and trace records. You can also filter log and trace records by using any extensions that are added by your application developers. You can use the `binaryLog` command-line tool to filter records based on the content of log and trace record extensions. For more information, see “BinaryLog command options” on page 625.

For example, to see all SIP application sessions that the SIP container processed, you can use the following `binaryLog` command:

```
binaryLog view binaryFile --includeExtension=SIPASId=* --format=advanced
```

### Developers

Developers can use binary logging to add custom extensions to log and trace records through the log record context API, `com.ibm.websphere.logging.hpel.LogRecordContext`. When binary logging stores log and trace records, it includes any extensions that are present in the log record context on the same thread. For example, you can write a servlet filter to add important HTTP request parameters to the log record context. While that servlet runs, the HPEL API adds those extensions to any log and trace records that are created on the same thread.

As with other log and trace record fields, developers can access the record extensions by using the HPEL API. This API is useful when you write tools to read from log and trace repositories. Developers can also use the log record context API to access extensions in custom log handlers, filters, and formatters at run time.

The following table describes the log and trace extensions, including the identifier that you can use to filter various aspects of the trace.

Table 113. Log and trace extensions.

Extension	Description
appName	Specifies the name of the Java Platform, Enterprise Edition (Java EE) application that the log or trace record relates to, if any.
requestID	Specifies the unique ID of the request that each log or trace record relates to, if any. For the application server to add the requestID extension to log and trace records, you must enable Cross Component Trace (XCT), also referred to in the administrative console as log and trace correlation. Request IDs are only added for certain types of requests, such as HTTP or JMS requests.

Table 113. Log and trace extensions (continued).

Extension	Description
SIPCallId	Specifies the SIP call identifier that is being processed by the SIP proxy server or SIP container. This information is common across SIP proxy servers and SIP containers. You can use this extension to track the SIP call flow across the various components. The SIP proxy server and SIP container automatically adds this identifier to each log and trace record when HPEL logging is enabled.
SIPASId	Specifies the SIP application session ID that is being processed by the SIP container. This information is common across SIP containers. You can use this extension to track the SIP call flow. The SIP container automatically adds this identifier to each log and trace record when HPEL logging is enabled.
SIPSessionId	Specifies the SIP session ID that is being processed by the SIP container. This information is common across SIP containers. You can use this extension to track the SIP call flow. The SIP container automatically adds this identifier to each log and trace record when HPEL logging is enabled.
SIPCallId2	<p>Specifies the second SIP call ID that is associated with the same SIP application session and is being processed by the SIP container. This information is common across SIP containers. You can use this extension to track the SIP call flow. The SIP container automatically adds this identifier to each log and trace record when HPEL logging is enabled.</p> <p>If more than two SIP call IDs are associated with a single SIP application session, only the first two IDs are recorded. The additional IDs are not recorded.</p>
SIPSessionId2	<p>Specifies the second SIP session ID that is associated with the same SIP application session and is being processed by the SIP container. This information is common across SIP containers. You can use this extension to track the SIP call flow. The SIP container automatically adds this identifier to each log and trace record when HPEL logging is enabled.</p> <p>If more than two SIP session IDs are associated with a single SIP application session, only the first two IDs are recorded. The additional IDs are not recorded.</p>
thread	Specifies the thread name of the request that each log or trace record relates to.

---

## Chapter 13. Reference

Reference information is organized to help you locate particular facts quickly.

---

### Programming Interfaces (APIs and SPIs)

Look up a package or class name to find details about programming interfaces for extending and interoperating with the systems management infrastructure of this product. This listing is in addition to the generated API and SPI documentation.

For Liberty API documentation, see the Programming Interfaces section in the WebSphere Application Server documentation.

---

### Messages

This reference information provides additional information about messages you might encounter while using the product. It is organized according to the identifier of the product feature that produces the message.

For Liberty messages, see the Messages section in the WebSphere Application Server documentation.



---

# Index

## A

- Alpha Liberty developer tools
  - installing 835
- Ant plug-in 967
- application defined data sources 1052
- application development
  - installing 1323
- Application Monitor
  - dynamic updates 975
- application security setting
  - restrictions 1480
- Application Server
  - creating 884
  - installing 835
  - logging 1461, 1467
  - runtime environments 943
- Application Server Tools feature
  - installing 835
- authentication
  - SPNEGO 596
- Authentication aliases 1300
- Authentication caches 1183
- Authentication configurations 1168
- authorization configurations 1254
- authorization on IBM i 1255
- authorization tasks 1254

## B

- basic registry security 1150
- Beta Liberty developer tools
  - installing 835

## C

- client authentication
  - SSL certificates 1164, 1298
- cnfiguration elements
  - jndiEntry 1336
- command port 897
- common OSGi bundles
  - sharing 1339
- common secure interoperability version 2 1277
- communications
  - securing 1147
- Configuration Admin Service
  - persisted identity 1119
- configuration data 1119
- configuration elements
  - application-bnd 1254
  - authData 1044
  - basicRegistry 1168, 1180
  - bundleRepository 1339
  - connectionManager 1055
  - context-root 1344
  - Embedding and Ref tags 972
  - Embedding only 972
  - httpSession 990
  - interceptors 1205

- configuration elements (*continued*)
  - jaasLoginContextEntry 1184
  - jaasLoginModule 1184
  - jndiReferenceEntry 1337
  - keystore 1153
  - keystores 1161
  - ldapRegistry 1169
  - ltpa 1188
  - pluginConfiguration 985
  - quickStartSecurity 1023
  - repertoire 1164, 1298
  - scim 1174
  - security-role 1254
  - SSL 1153
  - SSLDefault 1153
  - trustAssociation 1205
  - webAppSecurity 1203, 1297
  - webContainer 1012, 1015, 1017, 1019, 1058, 1060, 1067, 1068, 1070
- configuration instances 1122
- connection pooling 1055
- connectionManager properties 684
- context-root
  - deploying 1329
- CSIv2 1277
- custom configuration Liberty servers
  - create 1146
- Custom user registry 1318
- Custom user repository 1181

## D

- data source properties 683
- data source types 1047
- data sources
  - restrictions
    - runtime environments 1480
- Declarative services registration 1116
- default keystores 1164, 1298
- Derby data sources migration 685, 687
- Design view
  - restrictions 1488
- Developer Tools
  - installing 835
- diagnosis information 961
- directory locations and properties 894
- dynamic content 1141
- dynamic updates
  - controlling 975

## E

- EAR files
  - securing 1254
- EAR files
  - restrictions 1480
- Eclipse IDE
  - installing 835
- Eclipse Marketplace
  - installing 835

- Embedded server SPI 1145
- enterprise bundles
  - restrictions 1480
- entity manager
  - configuring 1347
- environment variables 947
- Equinox OSGi console
  - administering 937

## F

- factory configurations for default instances 1124
- feature configurations
  - features 482
- Feature definition format 1097
- feature examples
  - creating 1095
  - locating 1139
- feature FFDC records 1134
- feature logging SPI 1132
- feature restrictions
  - restrictions 1480
- feature SPIs 1095
- features installation 1143
- FFDC log files
  - logging 1461, 1467
  - restrictions 1480
- FFDC services
  - logging 1461, 1467
- FIPS support 1165

## H

- hashtable login module 1305
- HTTP access log 1444
- HTTP authentications
  - Trust Association Interceptors 1205
- HTTP port
  - configuring 938, 1186
- HTTP servlets
  - features 483
- httpEndpoint SSL configuration 1167
- HTTPS listener
  - features 483
- HTTPS port
  - features 483
  - restrictions 1480

## I

- Identity assertion customization 1316
- include statements 969
- installation integrity 875
- installation procedure
  - installing 835
- IP addresses
  - bootstrap properties 897

## J

- JAAS authentications
  - login modules 1184
- JAAS custom login modules
  - developing 1305
- Java command
  - configuration variables 970
- JavaMail
  - Administering 1074
- JAX-RS
  - context objects 1365
  - secure downstream resources 1355
- JAX-RS 2.0 1351, 1364
- JAX-RS applications
  - deploying 1348
- JDBC applications
  - deploying 1339
- JDBC data source
  - restrictions
    - runtime environments 1480
- JDBC data source
  - restrictions
    - tools 1488
- JDBC driver configurations 683
- JDBC driver location
  - deploying 1339
- JDBC drivers
  - deploying 1339
- JDBC tracing options 1342
- JDBC vendor properties 1047
- JMX connectors 1021, 1022, 1024, 1296
- JMX MBeans 1025
- JNDI binding 1140
- JNDI binding for constants 1336
- JNDI binding for dynamic values 1337
- JPA annotations
  - restrictions 1480
- JPA applications
  - configuring 1347
- JSF framework
  - features 483
- JSF support
  - restrictions 1480
- JVM monitoring 1423
- JVM options
  - bootstrap properties 897

## L

- Liberty 862, 1351, 1364
  - installing 835
  - OSGi console 968
  - restrictions 1480
- Liberty
  - administering 937
  - OSGi console 968
- Liberty feature project
  - specifying API and SPI packages 1107
- Liberty feature projects
  - adding OSGi bundles 1107
  - creating 1106
- Liberty features 1106, 1107
  - installing 1108, 1109
- Liberty features
  - administering 968

- Liberty features (*continued*)
  - deploying 1339
- Liberty profile
  - adding 1331
  - development environments 1323
  - global handlers 1134
  - monitoring local files 1132
- Liberty Profile
  - creating 884
  - exploring 943
- Liberty runtime environment
  - overview 934
- Liberty server
  - dumping 961
  - packaging 962, 1334
- Local download
  - creating 884
- local JMX connectors 1021
- Location services 1131
- log files
  - restrictions 1480
- logging configuration 1461, 1467
- LTPA keys 1188

## M

- MBeans attributes and operations 1029
- Merged Configuration view
  - configuring 944
- merged view
  - configuring 944

## O

- OAuth tasks 1257
- OSGi applications 981
  - deploying 1339
  - features 483
  - OSGi application integration 1323
- OSGi applications
  - deploying 1329
  - development environments 1323
  - running 1331
  - runtime environments 1480
- OSGi blueprint container specification
  - features 483
- OSGi bundles
  - adding to a Liberty feature project 1107
  - simple activation 1111
- OSGi Configuration Admin specification
  - persisted identities 1119
- OSGi console
  - administering 937
  - using 968
- OSGi console port
  - using 968
- OSGi metatype service extensions 1124
- OSGi Metatype Service specification
  - configuration metatypes 1121
- OSGi services availability 1115
- OSGi services registration 1112, 1113
- OSGi tool
  - installing 835

## P

- password encryption
  - encoding 1162
- plugin-cfg.xml in Liberty 985
- Port numbers
  - Default port numbers 899
- product extensions
  - features 577
- Project Explorer view
  - creating 884
- protected features 1137

## R

- reference tags 972
- role-based authorization 1254
- runtime environments
  - creating 884
  - exploring 943
- Runtime Explorer
  - exploring 943

## S

- SAF registry
  - features 483
- Schema Reference
  - server configurations 945
- secure communications 1151, 1277
- Security 1321
- Security differences 615
- security features 1147
- security infrastructure extensions 1301
- securityUtility command 1162
- Server Configuration editor
  - displaying 944
- server configuration files
  - logging 1461, 1467
- server configurations
  - include statements 969
- server configurations
  - configuring 938, 1186
  - creating 884
  - displaying 944
  - logging 1461, 1467
  - schema 945
- Server page 884
- server ports 897
- Server Version
  - logging 1461, 1467
- Servers view
  - configuring 1331
  - creating 884
  - editing 938
  - exploring 943
  - restrictions 1488
  - schema 945
  - server configurations 944
- serving environment
  - installing 835
- servlets loading 1012, 1015, 1017, 1019, 1058, 1060
- session failover 990
- Session Initiation Protocol (SIP)
  - Deploying SIP applications 1346
  - Troubleshooting 1498



- shared state variables 1305
- single sign-on
  - HTTP requests 596
  - LTPA cookies 1203, 1297
- SIP
  - troubleshooting 1495
- SSL certificate
  - creating 1162
- SSL certificates 1161
  - features 483
- SSL communications 1152, 1203
- SSL configuration attributes 1153

## T

- TAI configuration 1206
- TAI customization 1301
- TAI user feature 1141
- third-party features development 1095
- Thread pool monitoring 1425
- Tools
  - installing 835
- Tools feature
  - installing 835
- Transaction logs 1045
- transaction recovery 1044

- transaction service
  - administering 1043
- Troubleshooter reference
  - messages 1490
- Trust Association Interceptors 1205

## U

- usable callbacks 1305
- user ID
  - restrictions 1480

## V

- vendor databases configuration 1047

## W

- Web application monitoring 1424, 1441, 1442, 1443
- Web applications
  - features 483
- web applications
  - restrictions 1480
- web services applications
  - deploying 1348

- Web services monitoring 1176
- WebSphere Application Server Liberty
  - profile 943
- WebSphere Runtime Environment page
  - creating 884
- WebSphere Server page
  - creating 884

## X

- XMI files
  - securing 1254
- XMI files
  - restrictions 1480
- XML code
  - persisted identities 1119
- XML configuration files
  - bootstrap properties 897
- XML files
  - basic architecture 1
  - configuration metadata 1121
  - Liberty Repository 574
  - localizing 1130
  - server configurations 16, 1091, 1092
- XML snippets
  - features 968