

IBM WebSphere Application Server Network
Deployment, Version 5.0.2



Servers

Note

Before using this information, be sure to read the general information under “Trademarks and service marks” on page vii.

Compilation date: July 22, 2003

© Copyright International Business Machines Corporation 2002. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks and service marks vii

Chapter 1. Welcome to Servers. 1

Chapter 2. Configuring application servers 3

Application servers	3
Creating application servers	4
Configuring application servers for UTF-8 encoding	5
Managing Application Servers	5
Application server collection	6
Name.	6
Node	6
Status.	6
Application server settings.	6
Name.	7
Application Class loader Policy	7
Application Classloading Mode	7
Short name	7
Unique Id	7
Process ID	7
Cell Name	8
Node Name	8
State	8
End point collection	8
End Point Name	8
End point settings	8
End Point Name	8
Host	8
Port	9
Custom property collection	9
Name.	9
Value	9
Description	9
Required.	9
Valid Expression	9
Custom property settings	9
Name.	9
Value	9
Description	10
Server component collection.	10
Type.	10
Server component settings	10
Name	10
Initial State	10
Thread pool settings	10
Minimum size	10
Maximum size	11
Thread inactivity timeout.	11
Growable thread pool	11
Starting servers	11
Running an Application Server with a non-root user ID and the nodeagent as root	12
Running an Application Server and nodeagent with a non-root user ID	14

Detecting and handling problems with run-time components	16
Stopping servers.	16
Transports.	16
Configuring transports	17
HTTP transport collection	17
Host.	18
Port	18
SSL Enabled	18
HTTP transport settings	18
Host.	18
Port	18
SSL Enabled	18
SSL	18
Example: Setting custom properties for an HTTP transport	19
Custom services	20
Developing custom services	20
Custom service collection.	21
External Configuration URL.	21
Classname.	21
Display Name	21
Startup	22
Custom service settings	22
Startup	22
External Configuration URL.	22
Classname.	22
Display Name	22
Description	22
Classpath	23
Process definition	23
Defining application server processes.	23
Process definition settings	23
Start Command	23
Start Command Args	24
Stop Command	24
Stop Command Args	24
Terminate Command	24
Terminate Command Args	25
Executable Name	25
Executable Arguments.	25
Working Directory	25
Process execution settings	25
Process Priority	26
UMASK	26
Run As User	26
Run As Group	26
Run In Process Group	26
Process logs settings	26
Stdout File Name	26
Stderr File Name	27
Monitoring policy settings	27
Maximum Startup Attempts	27
Ping Interval	27
Ping Timeout.	27
Automatic Restart	28

Node Restart State	28
Java virtual machines (JVMs)	28
Using the JVM	28
Java virtual machine settings	29
Classpath	29
Boot Classpath	29
Verbose Class Loading	29
Verbose Garbage Collection	29
Verbose JNI	30
Initial Heap Size	30
Maximum Heap Size	30
Run HProf	30
HProf Arguments	30
Debug Mode	31
Debug Arguments	31
Generic JVM Arguments	31
Executable JAR File Name	32
Disable JIT	32
Operating System Name	32
Example: Configuring JVM sendRedirect calls to use context root	33
Preparing to host applications	33
Java memory tuning tips	34
Application servers: Resources for learning	38

Chapter 3. Managing Object Request Brokers 39

Object Request Brokers	39
Logical Pool Distribution (LPD)	40
Object Request Broker tuning guidelines	40
Object Request Broker service settings in administrative console	42
Request timeout	42
Request retries count	42
Request retries delay	42
Connection cache maximum	43
Connection cache minimum	43
ORB tracing	43
Locate request timeout	43
Force tunneling	43
Tunnel agent URL	44
Pass by reference	44
Object Request Broker service settings that can be added to the administrative console	45
com.ibm.CORBA.BootstrapHost	45
com.ibm.CORBA.BootstrapPort	45
com.ibm.CORBA.FragmentSize	46
com.ibm.CORBA.ListenerPort	46
com.ibm.CORBA.LocalHost	46
com.ibm.CORBA.ServerSocketQueueDepth	46
com.ibm.CORBA.ShortExceptionDetails	46
com.ibm.websphere.threadpool.strategy.implementation com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.calcinterval	47
com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.lruinterval	47
com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.outqueues	47
com.ibm.websphere.threadpool.strategy. .LogicalPoolDistribution.statsinterval	48

com.ibm.websphere.threadpool.strategy. .LogicalPoolDistribution.workqueue	48
Object Request Broker communications trace	48
Client-side programming tips for the Java Object Request Broker service	52
Character codeset conversion support for the Java Object Request Broker service	54
Object Request Brokers: Resources for learning	55

Chapter 4. Balancing workloads with clusters 57

Workload management (WLM)	57
Techniques for managing state	58
Clusters	59
Creating clusters	59
Server cluster collection	60
Server cluster settings	61
Creating cluster members	62
Cluster member collection	63
Member name	63
Node	63
Status	63
Cluster member settings	63
Replication	64
Replication entry	64
Replication domain	65
Replicating data	65
Internal replication domain collection	67
Name	67
Internal replication domain settings	67
Name	68
Request Timeout	68
Encryption Type	68
DRS Partition Size	68
Single Replica	69
Serialization Method	69
DRS Pool Size	69
DRS Pool Connections	70
Replicator entry collection	70
Replicator Name	70
Replicator entry settings	70
Replicator Name	70
Server	70
Replicator and Client Host Name	70
Replicator Port	71
Client Port	71
Starting clusters	71
Stopping clusters	72
Tuning a workload management configuration	72
Workload management run-time exceptions	73
Clustering and workload management: Resources for learning	74

Chapter 5. Web services gateway: Enabling Web services 75

Web services gateway - Frequently Asked Questions	75
Web services gateway - What is new in this release	76
Web services gateway - Completing the installation	77
Web services gateway - prerequisites and constraints	78

Establishing requirements for using a database with the gateway	79	Deploying Web services to the Web services gateway	106
Installing the gateway into a deployment manager cell	79	Data type representation - choosing between Generic classes and Deployed Java classes.	109
Installing the gateway into a stand-alone application server	82	Complex data types - mapping namespaces to packages	110
Testing the Web services gateway installation	86	Deploying Web services with Java bindings	110
Backing up and restoring a gateway configuration	86	Web services gateway - Supported types	111
Backing up and restoring UDDI publication links.	88	Publishing a Web service to a UDDI registry for deployment to the gateway.	112
Creating and updating a gateway cluster	88	Removing Web services from the Web services gateway	113
Administering the Web services gateway	90	Running the Web services gateway samples	114
Setting the namespace URI and WSDL URI for the Web services gateway	92	Passing SOAP messages with attachments through the Web services gateway	114
Working with channels	93	SOAP messages with attachments - a definition	115
Channels - entry points to the Web services gateway	93	Writing the WSDL extensions for SOAP messages with attachments	115
Listing and managing gateway-deployed channels	94	Administering security for the Web services gateway	116
Deploying channels to the Web services gateway	95	Enabling Web Services Security (WS-Security) for the gateway	116
Web services gateway - Channel deployment details	96	The Web services gateway and WS-Security	117
Removing channels from the Web services gateway	96	Configuring the gateway security bindings	119
Working with filters	97	Editing the service security configuration	122
Filters - service interceptors for the Web services gateway	97	Editing the target service security configuration	124
Listing and managing gateway-deployed filters	97	Enabling basic authentication and authorization for the gateway	126
Deploying filters to the Web services gateway	98	Enabling gateway-level authentication	126
Removing filters from the Web services gateway	98	Enabling operation-level authorization	128
Working with UDDI references	99	Operation-level security - role-based authorization	130
UDDI registries - Web service directories that integrate with the Web services gateway.	99	Invoking Web services over HTTPS	130
Listing and managing gateway-deployed UDDI references	101	Enabling proxy authentication for the gateway	131
Deploying UDDI references to the Web services gateway	102	Web services gateway troubleshooting tips	133
Removing UDDI references from the Web services gateway	103	Web services gateway messages	136
Working with Web services.	103	Web services gateway: Resources for learning	148
Listing and managing gateway-deployed Web services	104		

Trademarks and service marks

The following terms are trademarks of IBM Corporation in the United States, other countries, or both:

- Everyplace
- iSeries
- IBM
- Redbooks
- ViaVoice
- WebSphere
- zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product and service names may be trademarks or service marks of others.

Chapter 1. Welcome to Servers

The product provides application servers and more.

Application servers

Application servers extend the ability of a Web server to handle Web application requests. An application server enables a server to generate a dynamic, customized response to a client request.

Application servers use an Object Request Broker (ORB) for RMI/IIOP communication.

Java Messaging (JMS) servers

The product supports asynchronous messaging based on the Java Messaging Service (JMS) of a JMS provider that conforms to the JMS specification version 1.0.2 and supports the Application Server Facility (ASF) function defined within that specification.

For IBM WebSphere Application Server, the JMS functions (of the JMS provider) for an application server are served by the JMS server within the application server.

Chapter 2. Configuring application servers

An application server configuration provides settings that control how an application server provides services for running enterprise applications and their components.

This section describes how to create and configure application servers, and how to otherwise handle server configurations.

A WebSphere Application Server administrator can configure one or more application servers and perform tasks such as the following:

Steps for this task

1. Create application servers.
2. Manage application servers.
3. **(Optional)** Configure transports.
4. **(Optional)** Develop custom services.
5. **(Optional)** Define processes for the application server. As part of defining processes, you can define monitoring policies to track the performance of a process and name-value pairs for properties.
6. **(Optional)** Use the Java virtual machine.

After preparing a server, deploy an application or component on the server. See ["Preparing to host applications"](#) for a sample procedure that you might follow in configuring the application server run-time and resources.

Application servers

Application servers extend a Web server's capabilities to handle Web application requests, typically using Java technology. An application server makes it possible for a server to generate a dynamic, customized response to a client request.

For example, suppose—

1. A user at a Web browser on the public Internet visits a company Web site. The user requests to use an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing resources not handled directly by the Web server (such as servlets). It forwards the request to a WebSphere Application Server product.
4. The WebSphere Application Server product forwards the request to one of its application servers on which the application is running.
5. The invoked application then processes the user request. For example:
 - An application servlet prepares the user request for processing by an enterprise bean that performs the database access.
 - The application produces a dynamic Web page containing the results of the user query.
6. The application server collaborates with the Web server to return the results to the user at the Web browser.

The WebSphere Application Server product provides multiple application servers that can be either separately configured processes or nearly identical clones.

Creating application servers

You can create a new application server using the wsadmin tool or the Create New Application Server page of the administrative console. The steps below describe how to use the Create New Application Server page.

Steps for this task

1. Go to the Application Servers page and click **New**. This brings you to the Create New Application Server page.
2. Follow the instructions on the Create New Application Server page and define your application server.
 - a. Select a node for the application server.
 - b. Type in a name for the application server. The name must be unique within the node.
 - c. Select whether the new server will have unique ports for each HTTP transport. By default, this option is enabled. If you select this option, you might need to update the alias list for the virtual host that you plan to use with this server to contain these new port values. If you deselect this option, ensure that the default port values do not conflict with other servers on the same physical machine.
 - d. Select a template to be used in creating the new server. You can use a default application server template for your new server or use an existing application server as a template. The new application server will inherit all properties of the template server.
 - e. If you create the new server using an existing application server as a model, select whether to map applications from the existing server to the new server. By default, this option is disabled.
3. **(Optional)** To use multiple language encoding support in the administrative console, configure an application server with UTF-8 encoding enabled.

Results

The new application server appears in the list of servers on the Application Servers page.

What to do next

Note that the application server created has many default values specified for it. An application server has many properties that can be set and creating an application server on the Create New Application Server page specifies values for only a few of the important properties. To view all of the properties of your application server and to customize your application server further, click on the name of your application server on the Application Servers page and change the settings for your application server as needed.

Configuring application servers for UTF-8 encoding

To use multiple language encoding support in the administrative console, you must configure an application server with UTF-8 encoding enabled.

Steps for this task

1. Create an application server or use an existing application server.
2. On the Application Server page, click on the name of the server you want enabled for UTF-8.
3. On the settings page for the selected application server, click **Process Definition**.
4. On the Process Definition page, click **Java Virtual Machine**.
5. On the Java Virtual Machine page, specify `-Dclient.encoding.override=UTF-8` for **Generic JVM Arguments** and click **OK**.
6. Click **Save** on the console taskbar.
7. Restart the application server.

Note that the `autoRequestEncoding` option does not work with UTF-8 encoding enabled. The default behavior for WebSphere Application Server is, first, to check if `charset` is set on content type header. If it is, then the product uses content type header for character encoding; if it is not, then the product uses character encoding set on server using the system property `default.client.encoding`. If `charset` is not present and the system property is not set, then the product uses ISO-8859-1. Enabling `autoRequestEncoding` on a Web module changes the default behavior: if `charset` is not present on an incoming request header, the product checks the `Accept-Language` header of the incoming request and does encoding using the first language found in that header. If there is no `charset` on content type header and no `Accept language` header, then the product uses character encoding set on server using the system property `default.client.encoding`. As with the default behavior, if `charset` is not present and the system property is not set, then the product uses ISO-8859-1.

Managing Application Servers

To view information about an Application Server, use the Application Servers page. For the Network Deployment product, you can also use the Application Servers page to manage Application Servers. For the base WebSphere Application Server product, you cannot manage Application Servers from the administrative console; you must manage Application Servers from a console hosted by a Network Deployment deployment manager (dmgr) process. From the base product or the Network Deployment product, use the `wasadmin` tool or command line tools such as `as` and `asadmin` to manage the Application Server.

Steps for this task

1. Access the Application Servers page. Click **Servers > Application Servers** in the console navigation tree.
2. View information about Application Servers.

The Application Servers page lists Application Servers in the cell and the nodes holding the Application Servers.

To view additional information about a particular Application Server or to further configure an Application Server, click on the Application Server name under **Name**. This accesses the settings page for an Application Server.

To view product information for an Application Server:

- a. Verify that the Application Server is running.
- b. Display the **Runtime** tab on the settings page for an Application Server.
- c. Click **Product Information**.

The Product Information page displayed lists the WebSphere Application Server products installed for the Application Server, the version and build levels for the products, the build dates, and any interim fixes applied to the Application Server.

3. Create an Application Server. Click **New** and follow the instructions on the Create New Application Server page.
4. Monitor the running of Application Servers.
5. **(Optional)** Delete an Application Server.
 - a. Click **Servers > Application Servers** in the console navigation tree to access the Application Servers page.
 - b. Place a checkmark in the check box beside an Application Server to delete it.
 - c. Click **Delete**.
 - d. Click **OK** to confirm the deletion.

Application server collection

Use this page to view information about and manage application servers.

The Application Servers page lists application servers in the cell and the nodes holding the application servers.

To view this administrative console page, click **Servers > Application Servers**.

Name

Specifies a logical name for the server. Server names must be unique within a node.

Node

Specifies the name of the node for the application server.

Status

Indicates whether the application server is started or stopped.

Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.

Application server settings

Use this page to view or change the settings of an application server instance.

To view this administrative console page, click **Servers > Application Servers > *server_name***.

The **Configuration** tab provides editable fields and the **Runtime** tab provides read-only information. The **Runtime** tab is available only when the server is running.

Name

Specifies a logical name for the server. Server names must be unique within a node.

Data type	String
Default	server1

Application Class loader Policy

Specifies whether to use a single class loader to load all applications or to use a different class loader for each application.

The options are SINGLE and MULTIPLE. The default is to use a separate class loader for each application (MULTIPLE).

Data type	String
Default	MULTIPLE

Application Classloading Mode

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for JDK class loaders and WebSphere class loaders is PARENT_FIRST. By specifying PARENT_LAST, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or LinkageErrors if you have mixed use of overridden classes and non-overridden classes.

The options are PARENT_FIRST and PARENT_LAST. The default is to search in the parent class loader before searching in the application class loader to load a class.

Data type	String
Default	PARENT_FIRST

Short name

Specifies the short name of the server.

The name is 1-8 characters, alpha-numeric or national language. It cannot start with a numeric.

The system assigns a cell-unique, default short name.

Unique Id

Specifies the unique ID of this server.

The unique ID property is read only. The system automatically generates the value.

Process ID

Specifies a string identifying the process.

Data type	String
-----------	--------

Cell Name

Specifies the name of the cell for the application server.

Data type	String
Default	<i>host_name</i> Network

Node Name

Specifies the name of the node for the application server.

Data type	String
-----------	--------

State

Indicates whether the application server is started or stopped.

Data type	String
Default	Started

End point collection

Use this page to view and manage communication end points used by run-time components running within a process. End points provide host and port specifications for a server.

To view this administrative console page, click **Servers > Application Servers > *server_name* > End Points**.

Note that this page displays only when you are working with end points for application servers.

End Point Name

Specifies the name of an end point. Each name must be unique within the server.

End point settings

Use this to view and change the configuration for a communication end point used by run-time components running within a process. An end point provides host and port specifications for a server.

End Point Name

Specifies the name of the end point. The name must be unique within the server.

Note that this field displays only when you are defining an end point for an application server.

Data type	String
-----------	--------

Host

Specifies the 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 (such as the naming service, administrative service, or JMS broker).

For example, if the host name is myhost, the fully qualified DNS name can be myhost.myco.com and the IP address can be 155.123.88.201.

Data type	String
Default	*

Port

Specifies the port for which the service is configured to accept client requests. The port value is used in conjunction with the host name.

Data type	Integer
Default	None

Custom property collection

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

Name

Specifies the name (or key) for the property.

Value

Specifies the value paired with the specified name.

Description

Provides information about the name-value pair.

Required

Specifies whether the value field requires a value. If this box is checked, you must provide a value.

Valid Expression

Custom property settings

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

Name

Specifies the name (or key) for the property.

Data type	String
-----------	--------

Value

Specifies the value paired with the specified name.

Data type	String
-----------	--------

Description

Provides information about the name-value pair.

Data type String

Server component collection

Use this page to view information about and manage server component types such as application servers, messaging servers, or name servers.

To view this administrative console page, click **Servers > Application Servers > *server_name* > Server Components**.

Type

Specifies the type of internal server.

Server component settings

Use this page to view or configure a server component instance.

To view this administrative console, click **Servers > Application Servers > *server_name* > Server Components > *server_component_name***.

Name

Specifies the name of the component.

Data type String

Initial State

Specifies the desired state of the component when the server process starts. The options are: *Started* and *Stopped*. The default is *Started*.

Data type String
Default Started

Thread pool settings

Use this page to configure a group of threads that an application server uses. Requests are sent to the server through any of the HTTP transports. A thread pool enables components of the server to reuse threads to eliminate the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, click **Servers > Manage Application Servers > *server_name* > ORB Service > Thread Pool**. (You can reach this page through more than one navigational route.)

Minimum size

Specifies the minimum number of threads to allow in the pool.

Data type Integer
Default 10

Maximum size

Specifies the maximum number of threads to allow in the pool.

If your Tivoli Performance Viewer shows the Percent Maxed metric to remain consistently in the double digits, consider increasing the Maximum size. The Percent Maxed metric indicates the amount of time that the configured threads are used. If there are several simultaneous clients connecting to the server-side ORB, increase the size to support up to 1000 clients.

Data type	Integer
Default	50
Recommended	50 (25 on Linux systems)

Thread inactivity timeout

Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

Data type	Integer
Units	Milliseconds
Default	3500

Growable thread pool

Specifies whether the number of threads can increase beyond the maximum size configured for the thread pool.

Data type	Boolean
Default	Not enabled (false)
Range	Valid values are Allow thread allocation beyond maximum thread size or Not enabled.

Starting servers

Starting a server starts a new server process based on the process definition settings of the current server configuration.

There are several options for starting an Application Server:

Steps for this task

1. **(Optional)** Use the to start an Application Server from the command line.
2. **(Optional)** Start an Application Server for tracing and debugging.

To start the Application Server with standard Java debugging enabled:

- a. Click **Servers > Application Servers** from the administrative console navigation tree. Then, click the Application Server whose processes you want to trace and debug, **Process Definition**, and **Java Virtual Machine**.
- b. On the Java Virtual Machine page, place a checkmark in the check box for the **Debug Mode** setting to enable the standard Java debugger. If needed, set debug arguments. Then, click **OK**.
- c. Save the changes to a configuration file.
- d. Stop the Application Server.

- e. Start the Application Server again as described previously.

Running an Application Server with a non-root user ID and the nodeagent as root

Use this task to configure an Application Server to run as non-root.

By default, WebSphere Application Server on UNIX platforms uses the root user ID to run Application Servers. You can use a non-root user ID to run Application Servers.

If global security is enabled, it is not recommended that the Local OS be used for user registry. In general, using the Local OS user registry requires that all processes run as root. Refer to "Local operating system user registries" (not in this document) for details.

Using a non-root user ID to run Application Servers can be done by setting all the Application Servers to run under the same operating system group. If running the WebSphere JMS provider, add the jmsserver server to the mqm group to allow jmsserver to start the message queue. If not running jmsserver, you can use a group other than mqm in the following steps:

Steps for this task

1. Log on as root.
2. Create the was1 user ID to be used to run the Application Server.
3. Add users root and was1 to the mqm group.
4. Reboot the machine.
5. Configure Application Server properties for the root and was1 users.

Use the administrative console to complete the following steps:

- a. Define the nodeagent to run as a root process.

Click **System Management > Node Agents > nodeagent** (*for the node*) > **Process Definition > Process Execution** and change these values:

Property	Value
Run As User	root
Run As Group	mqm
UMASK	002

- b. Define each Application Server to run as a was1 process. Substitute the name of each server for server1.

Click **Servers > Application Servers > server1 > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	was1
Run As Group	mqm
UMASK	002

- c. If running the WebSphere JMS provider, define the jmsserver process to run as a root process.

Click **JMS Servers > jmsserver (for the node) > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	root
Run As Group	mqm
UMASK	002

6. Save and synchronize.

7. Stop all servers, including the server1 and jmsserver servers.

Use the **stopserver** command:

```
stopserver server1
stopserver jmsserver
```

8. Stop the node.

Use the **stopnode** command:

```
stopnode
```

9. As root, use operating system tools to change file permissions.

The following examples assume that the WebSphere Application Server installation root directory is /opt/WebSphere/AppServer:

```
chgrp mqm /opt/WebSphere
chgrp mqm /opt/WebSphere/AppServer
chgrp -R mqm /opt/WebSphere/AppServer/config
chgrp -R mqm /opt/WebSphere/AppServer/logs
chgrp -R mqm /opt/WebSphere/AppServer/wstemp
chgrp -R mqm /opt/WebSphere/AppServer/installedApps
chgrp -R mqm /opt/WebSphere/AppServer/temp
chgrp -R mqm /opt/WebSphere/AppServer/tranlog
chgrp -R mqm /opt/WebSphere/AppServer/cloudscape50
chgrp -R mqm /opt/WebSphere/AppServer/cloudscape51
chgrp -R mqm /opt/WebSphere/AppServer/bin/DefaultDB
chmod g+w /opt/WebSphere
chmod g+w /opt/WebSphere/AppServer
chmod -R g+w /opt/WebSphere/AppServer/config
chmod -R g+w /opt/WebSphere/AppServer/logs
chmod -R g+w /opt/WebSphere/AppServer/wstemp
chmod -R g+w /opt/WebSphere/AppServer/installedApps
chmod -R g+w /opt/WebSphere/AppServer/temp
chmod -R g+w /opt/WebSphere/AppServer/tranlog
chmod -R g+w /opt/WebSphere/AppServer/cloudscape50
chmod -R g+w /opt/WebSphere/AppServer/cloudscape51
chmod -R g+w /opt/WebSphere/AppServer/bin/DefaultDB
```

10. Start the node, the jmsserver, and all Application Servers.

Start the nodeagent and the jmsserver from root. Start each Application Server from the was1 user.

11. If running the WebSphere JMS provider, verify that the MQ queue is running.

Run the **dspmq** command:

```
dspmq
```

The name of the queue is WAS_wasnode_jmsserver.

Results

You can start an Application Server from a non-root user.

Running an Application Server and nodeagent with a non-root user ID

By default, each base Application Server node on Linux and UNIX platforms uses the root user ID to run the nodeagent process, the jmsserver process, and all Application Server processes. You can run the nodeagent, the jmsserver, and all Application Server processes under the same non-root user and user group.

If global security is enabled, the user registry must not be Local OS. Using the Local OS user registry requires the nodeagent to run as root.

Using the same non-root user and user group gives the nodeagent process the operating system permissions to start all other server processes. If using the WebSphere JMS provider, the user group must be mqm for the jmsserver to start the message queue. If you are not using the WebSphere JMS provider, you can specify a user group other than mqm.

For the steps that follow, assume that:

- wasadmin is the user to run all servers
- wasnode is the node name
- wascell is the cell name
- mqm and mqbrkrs are user groups associated with the WebSphere JMS provider
- server1 is the Application Server
- /opt/WebSphere/Appserver is the installation root
- jmsserver exists because you are using the WebSphere JMS provider

To configure a user ID to run the nodeagent and all server processes, complete the following steps:

Steps for this task

1. Log on as root.
2. Create user wasadmin with primary group mqm.
Also add user wasadmin to group mqbrkrs if you are running the WebSphere JMS provider.
3. Reboot the machine.
4. Define the nodeagent to run as a wasadmin process.

Click **System Management > Node Agents > nodeagent (for the node) > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm
UMASK	002

5. Define each Application Server to run as a wasadmin process. Substitute the name of each server for server1.
Click **Servers > Application Servers > server1 > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm

Property	Value
UMASK	002

- If running the WebSphere JMS provider, define the `jmsserver` process to run as a `wasadmin` process.

Click **JMS Servers > jmsserver** (for the node) > **Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm
UMASK	002

- Save and synchronize.

- Stop all servers, including the `server1` and `jmsserver` servers.

Use the **stopserver** command:

```
stopserver server1
stopserver jmsserver
```

- Stop the node.

Use the **stopnode** command:

```
stopnode
```

- If running the WebSphere JMS provider, delete the default queue manager for the Application Server.

From the `install_root/bin` directory, run the **deletemq** command as root:

```
deletemq.sh wascell wasnode jmsserver
```

- If running the WebSphere JMS provider, create the WebSphere JMS provider queue manager and broker for the Application Server.

Run the **createmq** command as `wasadmin`:

```
createmq.sh /opt/WebSphere/AppServer wascell wasnode jmsserver
```

- As root, use operating system tools to change file permissions:

```
chgrp mqm /opt/WebSphere
chgrp mqm /opt/WebSphere/AppServer
chgrp -R mqm /opt/WebSphere/AppServer/config
chgrp -R mqm /opt/WebSphere/AppServer/logs
chgrp -R mqm /opt/WebSphere/AppServer/wstemp
chgrp -R mqm /opt/WebSphere/AppServer/installedApps
chgrp -R mqm /opt/WebSphere/AppServer/temp
chgrp -R mqm /opt/WebSphere/AppServer/tranlog
chgrp -R mqm /opt/WebSphere/AppServer/cloudscape50
chgrp -R mqm /opt/WebSphere/AppServer/cloudscape51
chgrp -R mqm /opt/WebSphere/AppServer/bin/DefaultDB
chmod g+w /opt/WebSphere
chmod g+w /opt/WebSphere/AppServer
chmod -R g+w /opt/WebSphere/AppServer/config
chmod -R g+w /opt/WebSphere/AppServer/logs
chmod -R g+w /opt/WebSphere/AppServer/wstemp
chmod -R g+w /opt/WebSphere/AppServer/installedApps
chmod -R g+w /opt/WebSphere/AppServer/temp
chmod -R g+w /opt/WebSphere/AppServer/tranlog
chmod -R g+w /opt/WebSphere/AppServer/cloudscape50
chmod -R g+w /opt/WebSphere/AppServer/cloudscape51
chmod -R g+w /opt/WebSphere/AppServer/bin/DefaultDB
```

- Log in as `wasadmin`.

- From `wasadmin`, run the `startNode` command to start the `nodeagent` process:

- ```
startnode
```
15. From wasadmin, run the startserver command to start the jmsserver and all Application Servers:

```
startserver jmsserver
startserver server1
```
  16. If running the WebSphere JMS provider, verify that the MQ queue is running:  
Run the **dspmq** command:

```
dspmq
```

The name of the queue is WAS\_wasnode\_jmsserver.

### Results

You can start an Application Server, the jmsserver, and the nodeagent from a non-root user.

---

## Detecting and handling problems with run-time components

You must monitor the status of run-time components to ensure that, once started, they remain operational as needed.

### Steps for this task

1. Regularly examine the status of run-time components.  
Another way is to browse messages displayed under **WebSphere Runtime Messages** in the WebSphere status area at the bottom of the console. The run-time event messages marked with a red X provide detailed information on event processing.
2. If an application stops running when it should be operational, examine the application's status on an Applications page and try restarting the application.
3. If the run-time components do not restart, re-examine the messages and read information on problem determination to help you to restart the components.

---

## Stopping servers

Stopping an Application Server stops a server process based on the process definition settings in the current Application Server configuration.

### Steps for this task

---

## Transports

A transport is the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. When a user at a Web browser requests an application, the request is passed to the Web server, then along the transport to the Web container.

Transports define the characteristics of the connections between a Web server and an application server, across which requests for applications are routed. Specifically, they define the connection between the Web server plug-in and the Web container of the application server.

Administering transports is closely related to administering WebSphere Application Server plug-ins for Web servers. Indeed, without a plug-in configuration, a transport configuration is of little use.



## The internal transport

The internal HTTP transport allows HTTP requests to be routed to the application server directly or indirectly through a Web server plug-in. By default, the internal HTTP transport listens for HTTP requests on port 9080 and for HTTPS requests on port 9443.

For example, use the URL `http://localhost:9080/snoop` to send requests to the snoop servlet on the local machine over HTTP and `https://localhost:9443/snoop` to send requests to the snoop servlet on the local machine over HTTPS.

At times, you might be able to configure the internal transport to use ports other than 9080 and 9443. The transport configuration is a part of the Web container configuration. To change the port number, you must adjust your virtual host alias and what you type into the Web browser.

---

## Configuring transports

You configure transports to specify:

- How to manage a set of connections. For example, to specify the number of concurrent requests to allow.
- Whether to secure the connections with SSL
- Host and IP information for the transport participants

### Steps for this task

1. Create an HTTP transport.
  - a. Ensure that virtual host aliases include port values for the new transport.
  - b. Go to the HTTP Transports page and click **New**.
  - c. On the settings page for an HTTP transport, specify values such as the transport's host name and port number, then click **OK**.
2. **(Optional)** Change the configuration for an existing transport.
  - a. Ensure that virtual host aliases include port values for the transport you are changing.
  - b. Go to the HTTP Transports page and click on the transport under **Host** whose configuration you want to change.
  - c. On the settings page for an HTTP transport, which might have the page title `DefaultSSLSettings`, change the specified values as needed, then click **OK**.
3. Regenerate the WebSphere plug-in for the Web server.

### What to do next

If the Web server is located on a machine remote from the application server, you might also need to perform special configuration tasks to redirect application requests from the Web server machine to the application server machine.

---

## HTTP transport collection

Use this page to view or manage HTTP transports. Transports provide request queues between WebSphere plug-ins for Web servers and Web containers in which the Web modules of applications reside. When you request an application in a Web browser, the request is passed to the Web server, then along the transport to the Web container.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Web Container > HTTP Transports**.

## Host

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be localhost.

## Port

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system. The port number must be unique for each application server instance on a given machine.

## SSL Enabled

Specifies whether to protect connections between the WebSphere plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

---

## HTTP transport settings

Use this page to view and configure an HTTP transport. The name of the page might be that of an SSL setting such as DefaultSSLSettings.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Web Container > HTTP Transports > *host\_name***.

## Host

Specifies the host IP address to bind for transport.

If the application server is on a local machine, the host name might be localhost.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Port

Specifies the port to bind for transport. Specify a port number between 1 and 65535. The port number must be unique for each application server on a given machine.

|           |            |
|-----------|------------|
| Data type | Integer    |
| Range     | 1 to 65535 |

## SSL Enabled

Specifies whether to protect connections between the WebSphere plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## SSL

Specifies the Secure Sockets Layer (SSL) settings type for connections between the WebSphere plug-in and application server. The options include one or more SSL settings defined in the Security Center; for example, DefaultSSLSettings, ORBSSLSettings, or LDAPSSLSettings.

|           |                                               |
|-----------|-----------------------------------------------|
| Data type | String                                        |
| Default   | An SSL setting defined in the Security Center |

---

## Example: Setting custom properties for an HTTP transport

WebSphere Application Server has several transport properties that are not shown in the settings page for an HTTP transport. They are as follows:

### ConnectionIOTimeout

Specifies the maximum number of seconds to wait when trying to read or process data during a request. Data type: Integer.

The default value is five seconds. This value determines how long to wait while receiving two subsequent data packets for the same HTTP request. For example, using the default ConnectionIOTimeout setting of five seconds, if an HTTP client sends two data packets spaced six seconds apart, the timeout will fire, and a `java.io.InterruptedIOException` is raised. This will terminate the HTTP request and it will not be read. The HTTP client will have to reissue the request.

### ConnectionKeepAliveTimeout

Specifies the maximum number of seconds to wait for the next request on a keep alive connection. Data type: Integer.

### MaxKeepAliveRequests

Specifies the maximum number of requests which can be processed on a single keep alive connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

You can set these properties on either the Web Container or HTTP Transport **Custom Properties** pages. When set on the Web container Custom Properties page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify values for these custom properties for a specific transport on the HTTP Transport **Custom Properties** page:

### Steps for this task

1. Access the settings page for transport properties:
  - a. In the console navigation tree, click **Servers > Application Servers > *server\_name* > Web Container > HTTP Transport**
  - b. Click on the **HOST** whose properties you want to set.
  - c. Under **Additional Properties** select **Custom Properties**.
  - d. On the Custom Properties page, click **New**.
2. On the settings page for a new property, enter the name of the transport property and the value to which you want it set. For example, if you want the transport to wait a maximum of 60 seconds when trying to read or write data during a request, enter `ConnectionIOTimeout` for name and `60` for value. Then click **OK**.
3. Click **Save** on the console taskbar and save the changes to the configuration.
4. Restart the server.
5. Regenerate the Web server plug-in.

---

## Custom services

A custom service provides the ability to plug into a WebSphere application server to define a hook point that runs when the server starts and shuts down.

A developer implements a custom service containing a class that implements a particular interface. The administrator configures the custom service in the administrative console, identifying the class created by the developer. When an application server starts, any custom services defined for the application server are loaded and the server run-time calls their initialize methods.

---

## Developing custom services

To define a hook point to be run when a server starts and shuts down, you develop a custom service class and then use the administrative console to configure a custom service instance for an application server. When the application server starts, the custom service starts and initializes.

### Steps for this task

1. Develop a custom service class that implements the `com.ibm.websphere.runtime.CustomService` interface.

The properties passed by the application server run-time to the initialize method can include one for an external file containing configuration information for the service (retrieved with `externalConfigURLKey`). In addition, the properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a `Properties` object.

There is a shutdown method for the interface as well. Both methods of the interface declare that they may throw an exception, although no specific exception subclass is defined. If an exception is thrown, the run-time logs it, disables the custom service, and proceeds with starting the server.

2. On the Custom Service page of the administrative console, click **New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server, supplying the name of the class implemented.

If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file in the `externalConfigURL` field. This file name is passed into your custom service class.

3. Stop the application server and then restart the server.
4. Check the application server to ensure that the initialize method of the custom service ran as intended. Also ensure that the shutdown method performs as intended when the server stops.

### Usage scenario

As mentioned above, your custom services class must implement the `CustomService` interface. In addition, your class must implement the initialize and shutdown methods. Suppose the name of the class that implements your custom service is `ServerInit`, your code would declare this class as shown below. The code below assumes that your custom services class needs a configuration file. It shows how to process the input parameter in order to get the configuration file. If your class does not require a configuration file, the code that processes `configProperties` is not needed.

```

public class ServerInit implements CustomService
{
/**
* The initialize method is called by the application server run-time when the
* server starts. The Properties object passed to this method must contain all
* configuration information necessary for this service to initialize properly.
*
* @param configProperties java.util.Properties
*/
 static final java.lang.String externalConfigURLKey =
 "com.ibm.websphere.runtime.CustomService.externalConfigURLKey";

 static String ConfigFileName="";

 public void initialize(java.util.Properties configProperties) throws Exception
 {
 if (configProperties.getProperty(externalConfigURLKey) != null)
 {
 ConfigFileName = configProperties.getProperty(externalConfigURLKey);
 }

 // Implement rest of initialize method
 }

/**
* The shutdown method is called by the application server run-time when the
* server begins its shutdown processing.
*
* @param configProperties java.util.Properties
*/
 public void shutdown() throws Exception
 {
 // Implement shutdown method
 }
}

```

---

## Custom service collection

Use this page to view a list of services available to the application server and to see whether the services are enabled. A custom service provides the ability to plug into a WebSphere application server and define code that runs when the server starts or shuts down.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Custom Services**.

### External Configuration URL

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, the value provides a fully-qualified path name to that configuration file. This file name is passed into your custom service class.

### Classname

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

### Display Name

Specifies the name of the service.

## Startup

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

---

## Custom service settings

Use this page to configure a service that runs in an application server.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Custom Services > *custom\_service\_name***.

## Startup

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts. To enable the service, place a checkmark in the check box.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## External Configuration URL

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file for the value. This file name is passed into your custom service class.

|           |        |
|-----------|--------|
| Data type | String |
| Units     | URL    |

## Classname

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

|           |                 |
|-----------|-----------------|
| Data type | String          |
| Units     | Java class name |

## Display Name

Specifies the name of the service.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Description

Describes the custom service.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Classpath

Specifies the class path used to locate the classes and JAR files for this service.

|           |            |
|-----------|------------|
| Data type | String     |
| Units     | Class path |

---

## Process definition

A process definition specifies the run-time characteristics of an application server process.

A process definition can include characteristics such as JVM settings, standard in, error and output paths, and the user ID and password under which a server runs.

---

## Defining application server processes

To enhance the operation of an application server, you can define command-line information for starting or initializing an application server process. Such settings define run-time properties such as the program to run, arguments to run the program, the working directory.

### Steps for this task

1. Go to the settings page for a process definition in the administrative console. Click **Servers > Application Servers** in the console navigation tree, click on an application server name and then **Process Definition**.
2. On the settings page for a process definition, specify the name of the executable to run, any arguments to pass when the process starts running, and the working directory in which the process will run. Then click **OK**.
3. **(Optional)** Specify monitoring policies to track the performance of a process.
4. **(Optional)** Specify name-value pairs for properties needed by the process definition.
5. Stop the application server and then restart the server.
6. Check the application server to ensure that the process definition runs and operates as intended.

---

## Process definition settings

Use this page to view or change settings for a process definition, which provides command-line information for a process.

### Start Command

Specifies the platform-specific command to launch the server process.

#### Control process

|           |                                         |
|-----------|-----------------------------------------|
| Data type | String                                  |
| Format    | START <i>control JCL procedure name</i> |
| Example   | START BBO5ACR                           |

#### Servant process

For the servant process, the value on the start command specifies the procedure name that Workload Manager (WLM) uses to start the servant process. This value is used only if the WLM Dynamic Application

Environment feature is installed.

|           |                                         |
|-----------|-----------------------------------------|
| Data type | String                                  |
| Format    | START <i>servant JCL procedure name</i> |
| Example   | START BBO5ASR                           |

## Start Command Args

Specifies any additional arguments required by the start command.

### Control process

|           |                                                                                                  |
|-----------|--------------------------------------------------------------------------------------------------|
| Data type | String                                                                                           |
| Format    | JOBNAME= <i>server short name</i> ,ENV= <i>cell short name.node short name.server short name</i> |
| Example   | JOBNAME=BBOS001,ENV=SY1.SY1.BBOS001                                                              |

### Servant process

|           |                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------|
| Data type | String                                                                                            |
| Format    | JOBNAME= <i>server short name</i> S,ENV= <i>cell short name.node short name.server short name</i> |
| Example   | JOBNAME=BBOS001S,ENV=SY1.SY1.BBOS001                                                              |

## Stop Command

Specifies the platform-specific command to stop the server process

Specify two commands in the field, one for the Stop command and one for the Immediate Stop (CANCEL) command.

|           |                                                                |
|-----------|----------------------------------------------------------------|
| Data type | String                                                         |
| Format    | STOP <i>server short name</i> ;CANCEL <i>server short name</i> |
| Example   | STOP BBOS001;CANCEL BBOS001                                    |

## Stop Command Args

Specifies any additional arguments required by the stop command.

Specify arguments for the Stop command and the Immediate Stop (CANCEL) command.

|           |                                                                  |
|-----------|------------------------------------------------------------------|
| Data type | String                                                           |
| Format    | <i>stop command arg string;immediate stop command arg string</i> |
| Example   | ;ARMRESTART                                                      |

**Note:** In this example, Stop has no arguments. Immediate Stop has the argument ARMRESTART. A semicolon precedes ARMRESTART.

## Terminate Command

Specifies the platform-specific command to terminate the server process



|           |                                |
|-----------|--------------------------------|
| Data type | String                         |
| Format    | FORCE <i>server short name</i> |
| Example   | FORCE BBOS001                  |

## Terminate Command Args

Specifies any additional arguments required by the terminate command.

The default is an empty string.

|           |                                     |
|-----------|-------------------------------------|
| Data type | String                              |
| Format    | <i>terminate command arg string</i> |
| Example   | ARMRESTART                          |

## Executable Name

Specifies the executable name of the process.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Executable Arguments

Specifies executable commands that run when the process starts.

For example, the executable target program might expect three arguments: *arg1 arg2 arg3*.

|           |                             |
|-----------|-----------------------------|
| Data type | String                      |
| Units     | Java command-line arguments |

## Working Directory

Specifies the file system directory in which the process will run.

This directory is used to determine the locations of input and output files with relative path names.

Passivated enterprise beans are placed in the current working directory of the application server on which the beans are running. Make sure the working directory is a known directory under the root directory of the WebSphere Application Server product.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

---

## Process execution settings

Use this page to view or change command-line information for starting or initializing a UNIX process.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Process Definition > Process Execution**.

## Process Priority

Specifies the operating system priority for the process. Only root users can change this value.

|           |                                                                                                |
|-----------|------------------------------------------------------------------------------------------------|
| Data type | Integer                                                                                        |
| Default   | 1000 for WebSphere Application Server on most operating systems. On OS/400, the default is 25. |

## UMASK

Specifies the user mask under which the process runs (the file-mode permission mask).

|           |         |
|-----------|---------|
| Data type | Integer |
|-----------|---------|

## Run As User

Specifies the user that the process runs as.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Run As Group

Specifies the group that the process is a member of and runs as.

On OS/400, the Run As Group setting is ignored.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Run In Process Group

Specifies a specific process group for the process. This process group is useful for such things as processor partitioning. A system administrator can assign a process group to run on, for example, 6 of 12 processors. The default (0) is not to assign the process to any specific group.

On OS/400, the Run In Process Group setting is ignored.

|           |         |
|-----------|---------|
| Data type | Integer |
| Default   | 0       |

---

## Process logs settings

Use this page to view or change settings for specifying the files to which standard out and standard error streams write.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Process Definition > Process Logs**.

## Stdout File Name

Specifies the file to which the standard output stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the runtime tab to select a file for viewing. View the file by clicking **View**.

Direct server output to the administrative console or to the process that launched the server, by either deleting the file name or specifying console on the configuration tab.

|           |                |
|-----------|----------------|
| Data type | String         |
| Units     | File path name |

## Stderr File Name

Specifies the file to which the standard error stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the runtime tab to select a file for viewing. View the file by clicking **View**.

|           |                |
|-----------|----------------|
| Data type | String         |
| Units     | File path name |

---

## Monitoring policy settings

Use this page to view or change settings that control how the node agent monitors and restarts a process.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Process Definition > Monitoring Policy**.

## Maximum Startup Attempts

Specifies the maximum number of times to attempt to start the application server before giving up.

|           |         |
|-----------|---------|
| Data type | Integer |
|-----------|---------|

## Ping Interval

Specifies the frequency of communication attempts between the parent process, such as the node agent, and the process it has spawned, such as an application server. Adjust this value based on your requirements for restarting failed servers. Decreasing the value detects failures sooner; increasing the value reduces the frequency of pings, reducing system overhead.

|           |         |
|-----------|---------|
| Data type | Integer |
|-----------|---------|

## Ping Timeout

When a parent process is spawning a child process, such as when a process manager spawns a server, the parent process pings the child process to see whether the child was spawned successfully. This value specifies the number of seconds that the parent process should wait (after pinging the child process) before assuming that the child process failed.

|           |         |
|-----------|---------|
| Data type | Integer |
|-----------|---------|

Units

Seconds

## Automatic Restart

Specifies whether the process should restart automatically if it fails. The default is to restart the process automatically.

Data type

Boolean

Default

true

## Node Restart State

Specifies the desired state for the process after the node completely shuts down and restarts. The options are: *STOPPED*, *RUNNING*, *PREVIOUS*. The default is *STOPPED*.

Data type

String

Default

STOPPED

Range

Valid values are STOPPED, RUNNING, or PREVIOUS.

---

## Java virtual machines (JVMs)

The Java virtual machine (JVM) is an interpretive computing engine responsible for executing the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

---

## Using the JVM

As part of configuring an application server, you might define settings that enhance your system's use of the Java virtual machine (JVM).

To view and change the JVM configuration for an application server's process, use the Java Virtual Machine page of the console or use wsadmin to change the configuration through scripting.

### Steps for this task

1. Access the Java Virtual Machine page.
  - a. Click **Servers > Application Servers** in the console navigation tree.
  - b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.
  - c. On the settings page for the selected application server, click **Process Definition**.
  - d. On the Process Definition page, click **Java Virtual Machine**.
2. On the Java Virtual Machine page, specify values for the JVM settings as needed and click **OK**.
3. Click **Save** on the console taskbar.
4. Restart the application server.

### Usage scenario

""Configuring application servers for UTF-8 encoding"" provides an example that involves specifying a value for the **Generic JVM Arguments** property on the Java Virtual Machine page to enable UTF-8 encoding on an application server. Enabling UTF-8 allows multiple language encoding support to be used in the administrative console.

""Example: Configuring JVM sendRedirect calls to use context root"" provides an example that involves defining a property for the JVM.

---

## Java virtual machine settings

Use this page to view and change the Java virtual machine (JVM) configuration for the application server's process.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Process Definition > Java Virtual Machine**.

### Classpath

Specifies the standard class path in which the Java virtual machine code looks for classes.

Enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.

|           |            |
|-----------|------------|
| Data type | String     |
| Units     | Class path |

### Boot Classpath

Specifies bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources. You can separate multiple paths by a colon (:) or semi-colon (;), depending on operating system of the node.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

### Verbose Class Loading

Specifies whether to use verbose debug output for class loading. The default is not to enable verbose class loading.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

### Verbose Garbage Collection

Specifies whether to use verbose debug output for garbage collection. The default is not to enable verbose garbage collection.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## Verbose JNI

Specifies whether to use verbose debug output for native method invocation. The default is not to enable verbose Java Native Interface (JNI) activity.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## Initial Heap Size

Specifies the initial heap size available to the JVM code, in megabytes.

Increasing the minimum heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

In general, increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory. After the heap begins swapping to disk, Java performance suffers drastically.

|           |                                           |
|-----------|-------------------------------------------|
| Data type | Integer                                   |
| Default   | 64 for OS/400, 50 for all other platforms |

## Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes.

Increasing the heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

In general, increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory. After the heap begins swapping to disk, Java performance suffers drastically. Therefore, set the maximum heap size low enough to contain the heap within physical memory.

|           |                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------|
| Data type | Integer                                                                                                                 |
| Default   | 0 for OS/400, 256 for all other platforms.<br>Keep the value low enough to avoid paging or swapping-out-memory-to-disk. |

## Run HProf

Specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings using the HProf Arguments setting. The default is not to enable HProf profiler support.

If you set the Run HProf property to true, then you must specify command-line profiler arguments as values for the HProf Arguments property.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## HProf Arguments

Specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.

HProf arguments are only required if the Run HProf property is set to true.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Debug Mode

Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.

If you set the Debug Mode property to true, then you must specify command-line debug arguments as values for the Debug Arguments property.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## Debug Arguments

Specifies command-line debug arguments to pass to the JVM code that starts the application server process. You can specify arguments when Debug Mode is enabled.

Debug arguments are only required if the Debug Mode property is set to true.

|           |                             |
|-----------|-----------------------------|
| Data type | String                      |
| Units     | Java command-line arguments |

## Generic JVM Arguments

Specifies command line arguments to pass to the Java virtual machine code that starts the application server process.

The following are optional command line arguments that you can use by entering them into the **General JVM Arguments** field:

- **-Xquickstart:** You can use this value for initial compilation at a lower optimization level than in default mode, and later, depending on sampling results, you can recompile to the level of the initial compile in default mode. Use quickstart for applications where early moderate speed is more important than longrun throughput. In some debug scenarios, test harnesses and short-running tools, it is possible to realize startup time gains between 15-20%.  
**-DCOPT\_NQREACHDEF** can improve startup by an additional 15%.
- **-Xverify:none:** When using this value, the class verification stage is skipped during class loading . By using **-Xverify:none** with the just in time (JIT) compiler enabled, startup time is improved by 10-15%.
- **-Xnoclassgc:** You can use this value to disable class garbage collection, making class reuse more available, and slightly improving performance. Class garbage collection is enabled by default, but it is recommended that you enable it. You can monitor garbage collection using the verbose:gc configuration setting because its output includes class garbage collection statistics.
- **-Xgcthreads:** You can use several garbage collection threads at one time, also known as *parallel garbage collection*. When entering this value in the **Generic JVM Arguments** field, also enter the number of processors that your machine has, for example, **-Xgcthreads= number\_of\_processors**. It is recommended that you use parallel garbage collection if your machine has more than one processor. This argument applies only to the IBM Developer Kit.

- **-Xnoccompactgc:** This value disables heap compaction which is the most expensive garbage collection operation. Avoid compaction in IBM Developer Kit 1.3. If you disable heap compaction, you eliminate all associated overhead. When entering this value in the **Generic JVM Arguments** field, also enter the number of processors that your machine has, for example, `-Xnoccompactgc=number_of_processors`.
- **-Xinitsh:** You can use this value to set the initial heap size where class objects are stored. The method definitions and static fields are also stored with the class objects. Although the system heap size has no upper bound, set the initial size so that you do not incur the cost of expanding the system heap size, which involves calls to the operating system memory manager. You can compute a good initial system heap size by knowing the number of classes loaded in the WebSphere product, which is about 8,000 classes, and their average size. Having knowledge of the application helps you include them in the calculation.
- **-Xmc:** The thread local heap size is a portion of the heap that is allocated exclusively for a thread. Because of the thread local heap size, the thread does not need to lock the entire heap when allocating objects. However, when the thread local heap is full, object allocation is done from the heap that needs synchronization. A good local cache size is critical to performance and requires knowledge of the application and its objects.
- **-Xml:** You can use this value to set the limit of an object size to allocate from the local cache. Objects that exceed the limit size need allocating in the regular heap. Allocate objects from the local cache as much as possible or the local cache depletes because it does not grow dynamically. If you know some objects are going to be very large, allocate them from the regular heap.

|           |                             |
|-----------|-----------------------------|
| Data type | String                      |
| Units     | Java command line arguments |

## Executable JAR File Name

Specifies a full path name for an executable JAR file that the JVM code uses.

|           |           |
|-----------|-----------|
| Data type | String    |
| Units     | Path name |

## Disable JIT

Specifies whether to disable the just in time (JIT) compiler option of the JVM code.

If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

|             |                     |
|-------------|---------------------|
| Data type   | Boolean             |
| Default     | false (JIT enabled) |
| Recommended | JIT enabled         |

## Operating System Name

Specifies JVM settings for a given operating system. When started, the process uses the JVM settings for the operating system of the node.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|



---

## Example: Configuring JVM sendRedirect calls to use context root

If the `com.ibm.websphere.sendredirect.compatibility` property is not set and your application servlet code has statements such as `sendRedirect("/home.html")`, your Web browser might display messages such as *Error 404: No target servlet configured for uri: /home.html*. To instruct the server not to use the Web server's document root and to use instead the Web application's context root for `sendRedirect()` calls, configure the JVM by setting the `com.ibm.websphere.sendredirect.compatibility` property to a true or false value.

### Steps for this task

1. Access the settings page for a property of the JVM.
  - a. Click **Servers > Application Servers** in the console navigation tree.
  - b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.
  - c. On the settings page for the selected application server, click **Process Definition**.
  - d. On the Process Definition page, click **Java Virtual Machine**.
  - e. On the Java Virtual Machine page, click **Custom Properties**.
  - f. On the Properties page, click **New**.
2. On the settings page for a property, specify a name of `com.ibm.websphere.sendredirect.compatibility` and either true or false for the value, then click **OK**.
3. Click **Save** on the console taskbar.
4. Stop the application server and then restart the application server.

---

## Preparing to host applications

The default application server and a set of default resources are available to help you begin quickly. Suppose you choose instead to configure a new server and set of resources. Here is what you need to do in order to set up a run-time environment to support applications.

### Steps for this task

1. Create an application server.
2. Create a virtual host.
3. Configure a Web container.
4. Configure an EJB container.
5. Create resources for data access.
6. Create a JDBC provider and data source.
7. Create a URL and URL provider.
8. Create a JMS destination, connection, and provider.
9. Create a JavaMail session.
10. Create resources for session support.
11. Configure a Session Manager.

---

## Java memory tuning tips

Enterprise applications written in the Java language involve complex object relationships and utilize large numbers of objects. Although, the Java language automatically manages memory associated with object life cycles, understanding the application usage patterns for objects is important. In particular, verify the following:

- The application is not over-utilizing objects
- The application is not leaking objects
- The Java heap parameters are set properly to handle a given object usage pattern

Understanding the effect of garbage collection is necessary to apply these management techniques.

### The garbage collection bottleneck

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application, especially when running on symmetric multiprocessing (SMP) server machines. The Java virtual machine (JVM) uses a parallel garbage collector to fully exploit an SMP during most garbage collection cycles where the Sun HotSpot 1.3.1 JVM has a single-threaded garbage collector. .

### The garbage collection gauge

You can use garbage collection to evaluate application performance health. By monitoring garbage collection during the execution of a fixed workload, you gain insight as to whether the application is over-utilizing objects. Garbage collection can even detect the presence of memory leaks.

You can monitor garbage collection statistics using the **verbose:gc** JVM configuration setting. The **verbose:gc** format is not standardized between different JVMs or release levels.

For this type of investigation, set the minimum and maximum heap sizes to the same value. Choose a representative, repetitive workload that matches production usage as closely as possible, user errors included.

To ensure meaningful statistics, run the fixed workload until the application state is steady. It usually takes several minutes to reach a steady state.

### Detecting over-utilization of objects

You can check if the application is overusing objects, by observing the counters for the JVM runtime. You have to set the **-XrunpmiJvmpiProfiler** command line option, as well as the JVM module maximum level in order to enable the Java virtual machine profiler interface (JVMPi) counters. The best result for the average time between garbage collections is at least 5-6 times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15% of its time in garbage collection.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck. The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target. If you can not optimize the application, adding memory, processors and clones might help. Additional memory allows each clone to maintain a reasonable heap size. Additional processors allow the clones to run in parallel.

### **Detecting memory leaks**

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal Out of Memory exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. This is especially true for leaking applications where the high workload accelerates the magnification of the leakage and a memory allocation failure occurs.

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list contains important conclusions about memory leaks:

- **Long-running test**

Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests can lead to false alarms. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.

- **Repetitive test**

In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the

actual memory usage by inserting `System.gc()` in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.

- **Concurrency test**

Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or unreleased references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when choosing which test cases to use for memory leak testing:

- A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.
- Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as `Vector` and `Hashtable` are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the `get` method of a `Hashtable` object does not remove its reference to the retrieved object.

Heap consumption indicating a possible leak during a heavy workload (the application server is consistently near 100% CPU utilization), yet appearing to recover during a subsequent lighter or near-idle workload, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when small objects (less than 512 bytes) are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction has been run.

To avoid heap fragmentation, turn on the `-Xcompactgc` flag in the JVM advanced settings command line arguments. The `-Xcompactgc` function verifies that each garbage collection cycle eliminates fragmentation. However, compaction is a relatively expensive operation. See [Heap compaction \(-Xnocompactgc\)](#) for more information.

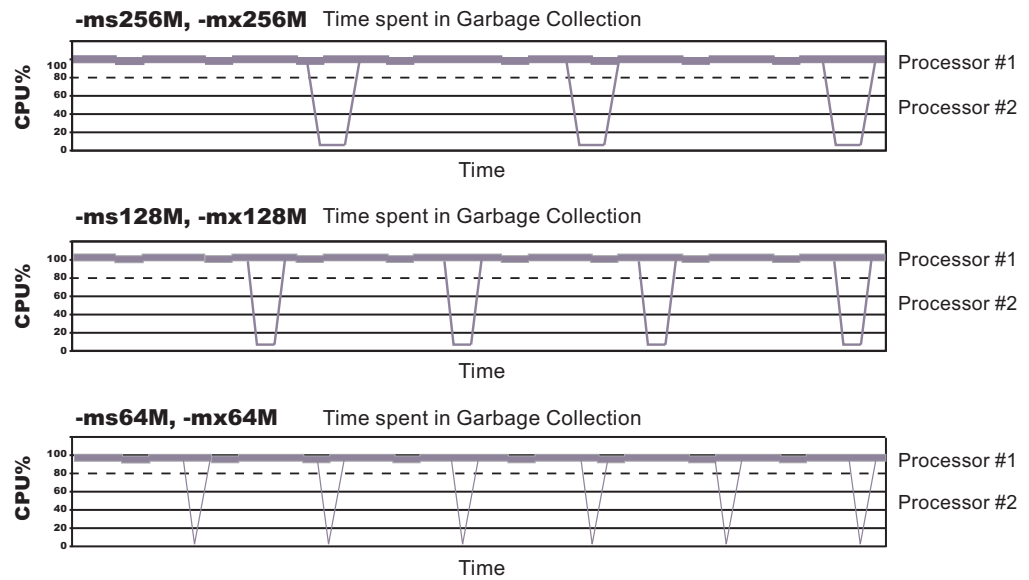
### **Java heap parameters**

The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer. See [Heap compaction](#) for more information.

*For performance analysis, the initial and maximum heap sizes should be equal.*

When tuning a production system where the working set size of the Java application is not understood, a good starting value for the initial heap size is 25% of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.

## Varying Java Heap Settings



The illustration represents three CPU profiles, each running a fixed workload with varying Java heap settings. In the middle profile, the initial and maximum heap sizes are set to 128MB. Four garbage collections occur. The total time in garbage collection is about 15% of the total run. When the heap parameters are doubled to 256MB, as in the top profile, the length of the work time increases between garbage collections. Only three garbage collections occur, but the length of each garbage collection is also increased. In the third profile, the heap size is reduced to 64MB and exhibits the opposite effect. With a smaller heap size, both the time between garbage collections and the time for each garbage collection are shorter. For all three configurations, the total time in garbage collection is approximately 15%. This example illustrates an important concept about the Java heap and its relationship to object utilization. There is always a cost for garbage collection in Java applications.

Run a series of test experiments that vary the Java heap settings. For example, run experiments with 128MB, 192MB, 256MB, and 320MB. During each experiment, monitor the total memory usage. If you expand the heap too aggressively, paging can occur. Use the `vmstat` command or the Windows NT or Windows 2000 Performance Monitor to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system. When all the runs are finished, compare the following statistics:

- Number of garbage collection calls
- Average duration of a single garbage collection call
- Ratio between the length of a single garbage collection call and the average time between calls

If the application is not over-utilizing objects and has no memory leaks, the state of steady memory utilization is reached. Garbage collection also occurs less frequently and for short duration.

If the heap free space settles at 85% or more, consider decreasing the maximum heap size values because the application server and the application are under-utilizing the memory allocated for heap.

---

## Application servers: Resources for learning


Use the following links to find relevant supplemental information about configuring application servers. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.



View links to additional information about:

- Programming instructions and examples
- Programming specifications
- Administration

### Programming instructions and examples

-  **WebSphere Application Server education**  
(<http://www.ibm.com/software/webservers/learn/>)

### Programming specifications

-  **The Java™ Virtual Machine Specification, Second Edition**  
(<http://java.sun.com/docs/books/vmspec/>)
-  **Sun's technology forum for the Java™ Virtual Machine Specification**  
(<http://forum.java.sun.com/forum.jsp?forum=37>)

### Administration

-  **Listing of all IBM WebSphere Application Server Redbooks**  
(<http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere>)

---

## Chapter 3. Managing Object Request Brokers

Default property values are set when the product is started and the Java Object Request Broker (ORB) service is initialized. These properties control the run-time behavior of the ORB and can also affect the behavior of product components that are tightly integrated with the ORB, such as security. It might be necessary to modify some ORB settings under certain conditions.

In every request/response exchange, there is a client-side ORB and a server-side ORB. It is important that the ORB properties be set for both sides as necessary.

After an ORB instance has been established in a process, changes to ORB properties do not affect the behavior of the running ORB instance. The process must be stopped and restarted in order for the modified properties to take effect.

The following steps are to be performed only as needed.

### Steps for this task

1. **(Optional)** Adjust timeout settings to improve handling of network failures.  
Before making these adjustments, be sure to read "ORB tuning guidelines."
2. If problems with the ORB arise, determine the problem.  
For help in troubleshooting, look at the ORB communications trace.

---

## Object Request Brokers

An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as if the remote objects were located in the same running process as the client, providing location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request on the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the stub, which, in turn, returns to the client application, as if the operation had been run locally.

This product uses an ORB to manage communication between client applications and server applications as well as communication among product components. During product installation, default property values are set when the ORB is initialized. These properties control the run-time behavior of the ORB and can also affect the behavior of product components that are tightly integrated with the ORB, such as security. This product does not support the use of multiple ORB instances.

---

## Logical Pool Distribution (LPD)

The Logical Pool Distribution (LPD) thread pool mechanism implements a strategy for improving the performance of requests that have shorter execution times.

The need for LPD is indicated by a mixture of EJB requests where the execution times vary across the request types, and the ORB thread pool must be constrained for performance reasons. In this case, longer execution time requests may tend to elongate the response times for shorter execution time requests by denying them adequate access to threads in the thread pool. LPD provides a mechanism allow shorter execution time requests greater access to execution threads.

LPD divides the Object Request Broker (ORB) thread pool into logical pools, as configured by the administrator using ORB custom properties starting with `com.ibm.websphere.threadpool.strategy.*`. The size of each pool is a percentage of the maximum number of ORB threads. The sum of the logical pool percentages should equal 100.

When LPD is active, incoming ORB requests are vectored to a pool based on historical execution time history for the request type. The request type is determined by the method which is qualified internally as unique across components. The LPD mechanism adjusts pool targets at runtime to optimize the distribution of requests across logical pools.

After it is enabled, the LPD mechanism can be tuned. Tuning exercises should be driven by response time and throughput measurements, as well as statistics produced by the LPD mechanism.

---

## Object Request Broker tuning guidelines

The following options exist for improving the performance of the Object Request Broker (ORB). Tuning results will vary among systems and applications.

- **Logical Pool Distribution (LPD) mechanism**

If you suspect that requests with longer execution times are elongating the response times for shorter execution time requests by denying them adequate access to threads in the thread pool, LPD provides a mechanism to allow the shorter requests greater access to execution threads.

For more information, see "Logical Pool Distribution (LPD)."

- **ORB timeout**

If Web clients that access Java applications running in the product environment are consistently experiencing problems with their requests, and the problem cannot be traced to other sources and addressed through other solutions, consider setting an ORB time-out value and adjusting it for your environment.

- Web browsers vary in their language for indicating that they have timed out. Usually, the problem is announced as a connection failure or no-path-to-server message.
- Aim to set an ORB time-out value to less than the time after which a Web client eventually times out. Because it can be difficult to tell how long Web clients wait before timing out, setting an ORB time-out requires experimentation. Another difficulty is that the ideal testing environment features some simulated network failures for testing the proposed setting value.
- Empirical results from limited testing indicate that 30 seconds is a reasonable starting value. Mainly, you need to ensure that the setting is not too low. To



fine-tune the setting, find a value that is not too low. Then gradually decrease the setting until reaching the threshold at which the value becomes too low. Set the value a little above the threshold.

- When an ORB time-out value is set too low, the symptom is numerous CORBA 'NO\_RESPONSE' exceptions, which occur even for some requests that should have been valid. If requests that should have been successful, for example, the server is not down, are being lost or refused, the value is likely to be too low.

**Note:** Do not adjust an ORB time-out value unless experiencing a problem, because configuring a value that is inappropriate for the environment can itself create a problem. If you set the value, experimentation might be needed to find the correct value for the particular environment. Configuring an incorrect value can produce results worse than the original problem.

You can adjust time-out intervals for the product's Java ORB through the following administrative settings:

- **Request timeout**, the number of seconds to wait before timing out on most pending ORB requests if the network fails
- **Locate request timeout**, the number of seconds to wait before timing out on a locate-request message

- **com.ibm.CORBA.numJNIReaders system property**

You can improve performance by setting the com.ibm.CORBA.numJNIReaders system property through a command-line script. This property specifies the number of threads to be shared for request handling when the native selector mechanism is enabled. The default value of this property is 2. Valid settings for this property range from 0 to 2147483647.

- **Determining the ORB message size**

The ORB breaks apart messages into fragments to send over the ORB connection. You can configure this fragment size through the com.ibm.CORBA.FragmentSize parameter.

To determine the size of the messages being transferred over the ORB and the number of fragments required to do so, you must first enable ORB tracing in the ORB Properties page in the webui and then enable ORBRas tracing from the logging and tracing page in the webui. You'll probably also want to bump up the trace file sizes as this can generate a lot of data. Restart the server and run at least one iteration (preferably several) of the case you wish to measure.

Then look at the traceable file and do a search for "Fragment to follow: Yes". This indicates that the ORB transmitted a fragment, but it still has at least one remaining fragment to send before the entire message has arrived. If No is indicated instead of Yes, this means that that particular fragment is the last in the entire message. It may also be the first if the message fit entirely into one fragment.

If you go to the spot where "Fragment to follow: Yes" is located, you will find a block that looks similar to this:

|                     |               |
|---------------------|---------------|
| Fragment to follow: | Yes           |
| Message size:       | 4988 (0x137C) |
| —                   |               |
| Request ID:         | 1411          |

This indicates that the amount of data in the fragment is 4988 bytes and the Request ID is 1411. Then if you do a search for all occurrences of "Request ID: 1411", you will see the number of fragments used to send that particular

message. If you add all the associated message sizes, you will have the total size of the message that's being send through the ORB.

---

## Object Request Broker service settings in administrative console

Use this page to configure the Java Object Request Broker (ORB) service.

To view this administrative console page, click **Servers > Application Servers > *serverName* > ORB Service**.

Several settings are available for controlling internal Object Request Broker (ORB) processing. You can use these settings to improve application performance in the case of applications containing enterprise beans. You can make changes to these settings for the default server or any application server configured in the administrative domain.

### Request timeout

Specifies the number of seconds to wait before timing out on a request message.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.RequestTimeout`.

|           |          |
|-----------|----------|
| Data type | int      |
| Units     | Seconds  |
| Default   | 180      |
| Range     | 0 to 300 |

### Request retries count

Specifies the number of times that the ORB attempts to send a request if a server fails. Retrying sometimes enables recovery from transient network failures.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.requestRetriesCount`.

|           |         |
|-----------|---------|
| Data type | int     |
| Default   | 1       |
| Range     | 1 to 10 |

### Request retries delay

Specifies the number of milliseconds between request retries.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.requestRetriesDelay`.

|           |              |
|-----------|--------------|
| Data type | int          |
| Units     | Milliseconds |
| Default   | 0            |
| Range     | 0 to 60      |

## Connection cache maximum

Specifies the largest number of connections allowed to occupy the connection cache for the service. If there are many simultaneous clients connecting to the server-side ORB, this parameter can be increased to support the heavy load up to 1000 clients.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.MaxOpenConnections`.

|           |             |
|-----------|-------------|
| Data type | Integer     |
| Units     | Connections |
| Default   | 240         |

## Connection cache minimum

Specifies the smallest number of connections allowed to occupy the connection cache for the service.

|           |             |
|-----------|-------------|
| Data type | Integer     |
| Units     | Connections |
| Default   | 100         |

## ORB tracing

Enables the tracing of ORB GIOP messages.

This setting affects two system properties: `com.ibm.CORBA.Debug` and `com.ibm.CORBA.CommTrace`. If you set these properties through command-line scripting, you must set both to `true` in order to enable the tracing of GIOP messages.

|           |                     |
|-----------|---------------------|
| Data type | Boolean             |
| Default   | Not enabled (false) |

## Locate request timeout

Specifies the number of seconds to wait before timing out on a `LocateRequest` message.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.LocateRequestTimeout`.

|           |          |
|-----------|----------|
| Data type | int      |
| Units     | Seconds  |
| Default   | 180      |
| Range     | 0 to 300 |

## Force tunneling

Controls how the client ORB attempts to use HTTP tunneling.

For direct access, the full name of this property is `com.ibm.CORBA.ForceTunnel`.

|           |                                                  |
|-----------|--------------------------------------------------|
| Data type | String                                           |
| Default   | NEVER                                            |
| Range     | Valid values are ALWAYS, NEVER, or WHENREQUIRED. |

Additional information about valid values follows:

**ALWAYS**

Use HTTP tunneling immediately, without trying TCP connections first.

**NEVER**

Disable HTTP tunneling. If a TCP connection fails, a CORBA system exception (COMM\_FAILURE) is thrown.

**WHENREQUIRED**

Use HTTP tunneling if TCP connections fail.

## Tunnel agent URL

Specifies the URL of the servlet used to support HTTP tunneling.

This must be a properly formed URL, such as `http://w3.mycorp.com:81/servlet/com.ibm.CORBA.services.IIOP TunnelServlet` or, for applets, `http://applethost:port/servlet/com.ibm.CORBA.services.IIOP TunnelServlet`. This field is required if **HTTP tunneling** is set.

For use in command-line scripting, the full name of this system property is `com.ibm.CORBA.TunnelAgentURL`.

## Pass by reference

Specifies how the ORB passes parameters. If enabled, the ORB passes parameters by reference instead of by value, which avoids making an object copy. If you do not enable pass by reference, the parameters are copied to the stack before every remote method call is made, which can be expensive.

If the EJB client and the EJB server are installed in the same WebSphere Application Server instance, and the client and server use remote interfaces, enabling Pass by reference can improve performance up to 50%. Pass by reference helps performance only where non-primitive object types are passed as parameters. Therefore, int and floats are always copied, regardless of the call model.

Enable this property with caution, because unexpected behavior can occur. If an object reference is modified by the remote method, the caller might change.

For use in command line scripting, the full name of this system property is `com.ibm.CORBA.iiop.noLocalCopies`.

|           |                     |
|-----------|---------------------|
| Data type | Boolean             |
| Default   | Not enabled (false) |

The use of this option for enterprise beans with remote interfaces violates the EJB Specification, Version 2.0 (see section 5.4). Object references passed to EJB methods or to EJB home methods are not copied and can be subject to corruption.

Consider the following example:

```
Iterator iterator = collection.iterator();
MyPrimaryKey pk = new MyPrimaryKey();
while (iterator.hasNext()) {
 pk.id = (String) iterator.next();
 MyEJB myEJB = myEJBHome.findByPrimaryKey(pk);
}
```

In this example, a reference to the same `MyPrimaryKey` object passes into WebSphere Application Server with a different ID value each time. Running this code with Pass by reference enabled causes a problem within the application server because multiple enterprise beans are referencing the same `MyPrimaryKey` object. To avoid this problem, set the system property `com.ibm.websphere.ejbcontainer.allowPrimaryKeyMutation` to true when Pass by reference is enabled. Setting Pass by reference to true causes the EJB container to make a local copy of the `PrimaryKey` object. As a result, however, a small portion of the performance advantage of setting Pass by reference is lost.

As a general rule, any application code that passes an object reference as a parameter to an enterprise bean method or EJB home method must be scrutinized to determine if passing that object reference results in loss of data integrity or other problems.

---

## Object Request Broker service settings that can be added to the administrative console

Use the Properties page to set and monitor settings associated with the Java Object Request Broker (ORB) service that do not appear on the main settings page by default.

To view the administrative console page, click **Servers > Application Servers > *serverName* > ORB Service > Custom Properties**.

To add properties to the page, click **New** and enter at least a name (case-sensitive) and value for the property. Then click **Apply**. When you are finished entering properties, click **OK**.

The page already might include Secure Sockets Layer (SSL) properties that were added during product setup. A list of additional properties associated with the Java ORB service follows.

### **com.ibm.CORBA.BootstrapHost**

Specifies the DNS host name or IP address of the machine on which initial server contact for this client resides. This setting is deprecated and will be removed in a future release.

For a command-line or programmatic alternative, see "Programming tips for the Java Object Request Broker service."

### **com.ibm.CORBA.BootstrapPort**

Specifies the port to which the ORB connects for bootstrapping. In other words, the port of the machine on which the initial server contact for this client is listening. This setting is deprecated and will be removed in a future release.

For a command-line or programmatic alternative, see "Programming tips for the Java Object Request Broker service."

Default

2809

## **com.ibm.CORBA.FragmentSize**

Specifies the size of GIOP fragments used by the ORB. If the total size of a request exceeds the set value, the ORB breaks up and sends multiple fragments until the entire request is sent. This should also be set on the client side with a -D system property if using a stand-alone java application.

Consider adjusting this when the amount of data that is sent over IOP exceeds 1k. Definitely adjust this if thread dumps show most client side threads stuck in a wait coming from writeLock. Sometimes a low amount of fragmentation is ok, so the parameter should be set such that most messages have few to no fragments.

|         |                                                                  |
|---------|------------------------------------------------------------------|
| Units   | Bytes.                                                           |
| Default | 1024                                                             |
| Range   | From 64 to largest value of Java int type that is divisible by 8 |

## **com.ibm.CORBA.ListenerPort**

Specifies the port on which this server listens for incoming requests. The setting of this property is valid only for client-side ORBs.

|         |                                            |
|---------|--------------------------------------------|
| Default | Next available system-assigned port number |
| Range   | 0 to 2147483647                            |

## **com.ibm.CORBA.LocalHost**

Specifies the host name or IP address of the system on which the server ORB is running. The setting of this property is valid only for client-side ORBs. Otherwise, the ORB obtains a value at run time by calling `InetAddress.getLocalHost().getHostAddress()`.

## **com.ibm.CORBA.ServerSocketQueueDepth**

Corresponds to the length of the TCP/IP stack listen queue and prevents WebSphere Application Server from rejecting requests when there is not space in the listen queue. If there are several simultaneous clients connecting to the server-side ORB, you can increase this parameter to support up to 1000 clients.

|         |                                               |
|---------|-----------------------------------------------|
| Default | 50                                            |
| Range   | From 50 to the largest value of Java int type |

## **com.ibm.CORBA.ShortExceptionDetails**

If set to any value, this specifies that the exception detail message that is returned whenever the server ORB encounters a CORBA system exception is to contain a short description of the exception as returned by the `toString()` method of `java.lang.Throwable`. Otherwise, the message contains the complete stack trace as returned by the `printStackTrace()` method of `java.lang.Throwable`.

## **com.ibm.websphere.threadpool.strategy.implementation**

If set to `com.ibm.ws.threadpool.strategy.LogicalPoolDistribution`, this enables the Logical Pool Distribution (LPD) thread pool strategy the next time you start the application server.

Some requests have shorter execution times than others. LPD is a mechanism for allowing these shorter requests greater access to execution threads. For more information, see "Logical Pool Distribution."

### **com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.calcinterval**

Specifies how often the Logical Pool Distribution (LPD) mechanism will readjust the pool execution target times. It cannot be turned off once this support is installed.

LPD must be enabled (see `com.ibm.websphere.threadpool.strategy.implementation`).

|           |                |
|-----------|----------------|
| Data type | Integer        |
| Units     | Milliseconds   |
| Default   | 30             |
| Range     | 20,000 minimum |

### **com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.lruinterval**

Specifies how long the Logical Pool Distribution internal data is kept for inactive requests. The mechanism tracks several statistics for each request type received. Consider removing requests that have not been active for a while.

LPD must be enabled (see `com.ibm.websphere.threadpool.strategy.implementation`).

|           |                           |
|-----------|---------------------------|
| Data type | Integer                   |
| Units     | Milliseconds              |
| Default   | 300,000 (5 minutes)       |
| Range     | 60,000 (1 minute) minimum |

### **com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.outqueues**

Specifies how many pools are created and how many threads are allocated to each pool in the Logical Pool Distribution mechanism

The ORB parameter for max threads controls the total number of threads. The outqueues parameter is specified as a comma separated list of percentages that should add up to 100. For example, the list 25,25,25,25 will set up 4 pools, each allocated 25% of the available ORB thread pool. The pools are indexed left to right from 0 to n-1. Each outqueue is dynamically assigned a target execution time by the calculation mechanism. Target execution times are assigned to outqueues in increasing order so pool 0 gets the requests with the least execution time and pool n-1 gets requests with the highest execution times.

LPD must be enabled (see `com.ibm.websphere.threadpool.strategy.implementation`).

|           |                                            |
|-----------|--------------------------------------------|
| Data type | Integers in comma separated list           |
| Default   | 25,25,25,25                                |
| Range     | Percentages in list must total 100 percent |

## **com.ibm.websphere.threadpool.strategy .LogicalPoolDistribution.statsinterval**

If active, statistics will be dumped to stdout after this interval expires, but only if requests have been processed. This keeps the mechanism from filling the log files with redundant information. These stats are beneficial for tuning the Logical Pool Distribution mechanism.

LPD must be enabled (see `com.ibm.websphere.threadpool.strategy.implementation`).

|           |                             |
|-----------|-----------------------------|
| Data type | Integer                     |
| Units     | Milliseconds                |
| Default   | 0 (meaning Off)             |
| Range     | 30,000 (30 seconds) minimum |

## **com.ibm.websphere.threadpool.strategy .LogicalPoolDistribution.workqueue**

Specifies the size of a new queue where incoming requests wait for dispatching. Pertains to the Logical Pool Distribution mechanism.

LPD must be enabled (see `com.ibm.websphere.threadpool.strategy.implementation`).

|           |            |
|-----------|------------|
| Data type | Integer    |
| Default   | 96         |
| Range     | 10 minimum |

---

## **Object Request Broker communications trace**

The Object Request Broker (ORB) communications trace, typically referred to as *CommTrace*, contains the sequence of General InterORB Protocol (GIOP) messages sent and received by the ORB during application execution. It might be necessary to understand the low-level sequence of client-to-server or server-to-server interactions during problem determination. This article uses trace entries from log examples to explain the contents of the log and help you understand the interaction sequence. It focuses only in the GIOP messages and does not discuss in detail additional trace information that appears when intervening with the GIOP-message boundaries.

### **Location**

When ORB tracing is enabled, this information is placed in `install_root/logs/trace`.

### **Usage notes**

- Is this file read-only?  
Yes
- Is this file updated by a product component?  
This file is updated by the administrative function.
- How and when are the contents of this file used?  
You use this file to localize and resolve ORB-related problems.

### **How to interpret the output**

The following sections refer to sample log output found later in this topic.



### Identifying information

The start of a GIOP message is identified by a line which contains either "OUT GOING:" or "IN COMING:" depending on whether the message is sent or received by the process that is being traced.

Following the identifying line entry is a series of items, formatted for convenience, with information extracted from the raw message that identify the endpoints in this particular message interaction. See lines 3-13 in both examples. The formatted items include the following:

- GIOP message type (line 3)
- Date and time that message was recorded (line 4)
- Information useful in uniquely identifying the thread in execution when the message was recorded, with other thread-specific information (line 5, broken for publication in the reply example)
- Local and remote TCP/IP ports used for the interaction (lines 6 through 9)
- GIOP version, byte order, whether the message is a fragment, and message size (lines 10 through 13)

### Request ID, response expected and reply status

Following the introductory message information, the request ID is an integer generated by the ORB. It is used to identify and associate each request with its corresponding reply. This is necessary because the ORB can receive requests from multiple clients and must be able to associate each reply with the corresponding originating request.

- Lines 15-17 in the request example show the request ID, followed by an indication to the receiving endpoint that a response is expected (CORBA allows sending of one-way requests for which a response is not expected.)
- Line 15 in Sample Log Entry - GIOP Reply shows the request ID; line 33 shows the reply status received after completing the previously sent request.

### Object Key

Lines 18-20 in the request example show the object key, the internal representation used by the ORB during execution to identify and locate the target object intended to receive the request message. Object keys are not standardized.

### Operation

Line 21 in the request example shows the name of the operation to be executed by the target object in the receiving endpoint. In this example, the specific operation requested is named `_get_value`.

### Service context information

The service contexts in the message are also formatted for convenience. Each GIOP message might contain a sequence of service contexts sent/received by each endpoint. Service contexts, identified uniquely with an ID, contain data used in the specific interaction, such as security, character codeset conversion, and ORB version information. The content of some of the service contexts is standardized and specified by OMG, while other service contexts are proprietary and specified by each vendor. IBM-specific service contexts are identified with IDs that begin with 0x4942.

Lines 22-41 in the request example illustrate typical service context entries. There are three service contexts in the request message, as shown in line

22. The ID, length of data, and raw data for each service context is printed next. Lines 23-25 show an IBM-proprietary context, as indicated by the ID 0x49424D12. Lines 26-41 show two standard service contexts, identified by ID 0x6 (line 26) and 0x1 (line 39).

Lines 16-32 in the Sample Log Entry - GIOP Reply illustrate two service contexts, one IBM-proprietary (line 17) and one standardized (line 20).

For the definition of the standardized service contexts, see the CORBA specification. Service context 0x1 (CORBA::IOP::CodeSets) is used to publish the character codesets supported by the ORB in order to negotiate and determine the codeset used to transmit character data. Service context 0x6 (CORBA::IOP::SendingContextRunTime) is used by RMI-IIOP to provide the receiving endpoint with the IOR for the SendingContextRuntime object. IBM service context 0x49424D12 is used to publish ORB PartnerVersion information in order to support release-to-release interoperability between sending and receiving ORBs.

#### Data offset

Line 42 in the request example shows the offset, relative to the beginning of the GIOP message, where the remainder body of the request or reply message is located. This portion of the message is specific to each operation and varies from operation to operation. Therefore, it is not formatted, as the specific contents are not known by the ORB. The offset is printed as an aid to quickly locating the operation-specific data in the raw GIOP message dump, which follows the data offset.

#### Raw GIOP message dump

Starting at line 45 in the request example and line 36 in Sample Log Entry - GIOP Reply, a raw dump of the entire GIOP message is printed in hexadecimal format. Request messages contain the parameters required by the given operation and reply messages contain the return values and content of output parameters as required by the given operation. For brevity, not all of the raw data has been included in the figures.

#### Sample Log Entry - GIOP Request

```
1. OUT GOING:

3. Request Message
4. Date: April 17, 2002 10:00:43 PM CDT
5. Thread Info: P=842115:0=1:CT
6. Local Port: 1243 (0x4DB)
7. Local IP: jdoe.austin.ibm.com/192.168.1.101
8. Remote Port: 1242 (0x4DA)
9. Remote IP: jdoe.austin.ibm.com/192.168.1.101
10. GIOP Version: 1.2
11. Byte order: big endian
12. Fragment to follow: No
13. Message size: 268 (0x10C)
--
15. Request ID: 5
16. Response Flag: WITH_TARGET
17. Target Address: 0
18. Object Key: length = 24 (0x18)
 4B4D4249 00000010 BA4D6D34 000E0008
 00000000 00000000
21. Operation: _get_value
22. Service Context: length = 3 (0x3)
23. Context ID: 1229081874 (0x49424D12)
24. Context data: length = 8 (0x8)
 00000000 13100003
26. Context ID: 6 (0x6)
```

```

27. Context data: length = 164 (0xA4)
 00000000 00000028 49444C3A 6F6D672E
 6F72672F 53656E64 696E6743 6F6E7465
 78742F43 6F646542 6173653A 312E3000
 00000001 00000000 00000068 00010200
 0000000E 3139322E 3136382E 312E3130
 310004DC 00000018 4B4D4249 00000010
 BA4D6D69 000E0008 00000000 00000000
 00000002 00000001 00000018 00000000
 00010001 00000001 00010020 00010100
 00000000 49424D0A 00000008 00000000
 13100003
39. Context ID: 1 (0x1)
40. Context data: length = 12 (0xC)
 00000000 00010001 00010100
42. Data Offset: 118

45. 0000: 47494F50 01020000 0000010C 00000005 GIOP.....
46. 0010: 03000000 00000000 00000018 4B4D4249 KMBI
47. 0020: [remainder of message body deleted for brevity]

```

### Sample Log Entry - GIOP Reply

```

1. IN COMING:

3. Reply Message
4. Date: April 17, 2002 10:00:47 PM CDT
5. Thread Info: RT=0:P=842115:O=1:com.ibm.rmi.transport.TCPTransportConnection
5a. remoteHost=192.168.1.101 remotePort=1242 localPort=1243
6. Local Port: 1243 (0x4DB)
7. Local IP: jdoe.austin.ibm.com/192.168.1.101
8. Remote Port: 1242 (0x4DA)
9. Remote IP: jdoe.austin.ibm.com/192.168.1.101
10. GIOP Version: 1.2
11. Byte order: big endian
12. Fragment to follow: No
13. Message size: 208 (0xD0)
--
15. Request ID: 5
16. Service Context: length = 2 (0x2)
17. Context ID: 1229081874 (0x49424D12)
18. Context data: length = 8 (0x8)
 00000000 13100003
20. Context ID: 6 (0x6)
21. Context data: length = 164 (0xA4)
 00000000 00000028 49444C3A 6F6D672E
 6F72672F 53656E64 696E6743 6F6E7465
 78742F43 6F646542 6173653A 312E3000
 00000001 00000000 00000068 00010200
 0000000E 3139322E 3136382E 312E3130
 310004DA 00000018 4B4D4249 00000010
 BA4D6D34 000E0008 00000001 00000000
 00000002 00000001 00000018 00000000
 00010001 00000001 00010020 00010100
 00000000 49424D0A 00000008 00000000
 13100003
33. Reply Status: NO_EXCEPTION

36. 0000: 47494F50 01020001 000000D0 00000005 GIOP.....
37. 0010: 00000000 00000002 49424D12 00000008 IBM.....
38. 0020: [remainder of message body deleted for brevity]

```

---

## Client-side programming tips for the Java Object Request Broker service

This article includes programming tips for applications that communicate with the client-side Object Request Broker (ORB) that is part of the Java ORB service.

### Resolution of initial references to services

Client applications can use the properties *ORBInitRef* and *ORBDefaultInitRef* to configure the network location that the Java ORB service uses to find a service such as naming. Once set, these properties are included in the parameters used to initialize the ORB, as follows:

```
org.omg.CORBA.ORB.init(java.lang.String[] args,
 java.util.Properties props)
```

You can set these properties in client code or by command-line argument. It is possible to specify more than one service location by using multiple *ORBInitRef* property settings (one for each service), but only a single value for *ORBDefaultInitRef* may be specified. For more information about the two properties and the order of precedence that the ORB uses to locate services, read the CORBA/IOP specification, cited in "Resources for learning."

For setting in client code, these properties are `com.ibm.CORBA.ORBInitRef.service_name` and `com.ibm.CORBA.ORBDefaultInitRef`, respectively. For example, to specify that the naming service (NameService) is located in `sample.server.com` at port 2809, set the `com.ibm.CORBA.ORBInitRef.NameService` property to `corbaloc::sample.server.com:2809/NameService`.

For setting by command-line argument, these properties are `-ORBInitRef` and `-ORBDefaultInitRef`, respectively. To locate the same naming service specified previously, use the following Java command (split here for publication only):

```
java program -ORBInitRef
 NameService=corbaloc::sample.server.com:2809/NameService
```

After these properties have been set for services supported by the ORB, J2EE applications obtain the initial reference to a given service by calling the `resolve_initial_references` function on the ORB as defined in the CORBA/IOP specification.

### Preferred API for obtaining an ORB instance

For J2EE applications, you can use either of the following approaches. However, it is strongly recommended that you use the JNDI approach to ensure that the same ORB instance is used throughout the client application; you will avoid the unintended inconsistencies that might occur when different ORB instances are used.

**JNDI approach:** For J2EE applications (including enterprise beans, J2EE clients and servlets), you can obtain an ORB instance by creating a JNDI InitialContext object and looking up the ORB under the name `java:comp/ORB`, as follows:

```
javax.naming.Context ctx = new javax.naming.InitialContext();
org.omg.CORBA.ORB orb =
 (org.omg.CORBA.ORB) javax.rmi.PortableRemoteObject.
 narrow(ctx.lookup("java:comp/ORB"), org.omg.CORBA.ORB.class);
```

The ORB instance obtained using JNDI is a singleton object, shared by all J2EE components running in the same Java virtual machine process.

**CORBA approach:** Because thin-client applications do not run in a J2EE container, they cannot use JNDI interfaces to look up the ORB. In this case, you can obtain an ORB instance by using CORBA programming interfaces, as follows:

```
java.util.Properties props = new java.util.Properties();
java.lang.String[] args = new java.lang.String[0];
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, props);
```

In contrast to the JNDI approach, the CORBA specification requires that a new ORB instance be created each time the ORB.init method is called. If necessary to change the ORB's default settings, you can add ORB property settings to the Properties object that is passed in the ORB.init() call.

The use of com.ibm.ejs.oa.EJSORB.getORBInstance(), supported in previous releases of this product, has been deprecated.

### API restrictions with sharing an ORB instance among J2EE application components

For performance reasons, it often makes sense to share a single ORB instance among components in a J2EE application. As required by the J2EE Specification, Version 1.3, all web and EJB containers provide an ORB instance in the JNDI namespace as java:comp/ORB. Each container can share this instance among application components but is not required to. For proper isolation between application components, application code must comply with the following restrictions:

- Do not call the ORB shutdown method
- Do not call org.omg.CORBA\_2\_3.ORB methods register\_value\_factory or unregister\_value\_factory

In addition, an ORB instance should not be shared among application components in different J2EE applications.

### Required use of rmic and idlj shipped with the IBM Developer Kit

The Java Runtime Environment (JRE) used by this product includes the tools **rmic** and **idlj**. You use the tools to generate Java language bindings for the CORBA/IIOP protocol.

During product installation, the tools are installed in the directory *installation\_root*/java/ibm\_bin, where *installation\_root* is the installation directory for the product. Versions of these tools included with Java development kits in \$JAVA\_HOME/bin other than the IBM Developer Kit installed with this product are incompatible with this product.

When you install this product, the directory *installation\_root*/java/ibm\_bin is included in the \$PATH search order to enable use of the rmic and idlj scripts provided by IBM. Because the scripts are in *installation\_root*/java/ibm\_bin instead of the JRE standard location *installation\_root*/java/bin, it is unlikely that you will overwrite them when applying maintenance to a JRE not provided by IBM.

In addition to the rmic and idlj tools, the JRE also includes Interface Definition Language (IDL) files. The files are based on those defined by the Object

Management Group (OMG) and can be used by applications that need an IDL definition of selected ORB interfaces. The files are placed in the *installation\_root/java/ibm\_lib* directory.

Before using either the *rmic* or *idlj* tool, ensure that the *installation\_root/java/ibm\_bin* directory is included in the proper PATH variable search order in the environment. If your application will use IDL files in the *installation\_root/java/ibm\_lib* directory, also ensure that the directory is included in the PATH variable.

---

## Character codeset conversion support for the Java Object Request Broker service

The CORBA/IIOP specification defines a framework for negotiation and conversion of character codesets used by the Java Object Request Broker (ORB) service. This product supports the framework and provides the following system properties for modifying the default settings:

### **com.ibm.CORBA.ORBCharEncoding**

Specifies the name of the native codeset that the ORB is to use for character data (referred to as NCS-C in the CORBA/IIOP specification). By default, the ORB uses UTF8. (In contrast, the default value for versions 3.5.x and 4.0.x of this product was ISO8859\_1, also known as Latin-1.) Valid codeset values for this property are shown in the table that follows this list; values that are valid only for ORBWCharDefault are indicated.

### **com.ibm.CORBA.ORBWCharDefault**

Specifies the default codeset that the ORB is to use for transmission of wide character data when no codeset for wide character data is found in the tagged component in the Interoperable Object Reference (IOR) or in the GIOP service context. If no codeset for wide character data is found and this property is not set, the ORB raises an exception, as specified in the CORBA specification. There is no default value set for this property. The only valid codeset values for this property are UCS2 or UTF16.

The CORBA codeset negotiation/conversion framework specifies the use of codeset registry IDs as defined in the Open Software Foundation (OSF) codeset registry. The ORB translates the Java *file.encoding* names shown in the following table to the corresponding OSF registry IDs. These IDs are then used by the ORB in the IOR Codeset tagged component and GIOP Codeset service context as specified in the CORBA/IIOP specification.

| Java name | OSF registry ID | Comments |
|-----------|-----------------|----------|
| ASCII     | 0x00010020      |          |
| ISO8859_1 | 0x00010001      |          |
| ISO8859_2 | 0x00010002      |          |
| ISO8859_3 | 0x00010003      |          |
| ISO8859_4 | 0x00010004      |          |
| ISO8859_5 | 0x00010005      |          |
| ISO8859_6 | 0x00010006      |          |
| ISO8859_7 | 0x00010007      |          |
| ISO8859_8 | 0x00010008      |          |
| ISO8859_9 | 0x00010009      |          |

| Java name       | OSF registry ID | Comments                       |
|-----------------|-----------------|--------------------------------|
| ISO8859_15_FDIS | 0x0001000F      |                                |
| Cp1250          | 0x100204E2      |                                |
| Cp1251          | 0x100204E3      |                                |
| Cp1252          | 0x100204E4      |                                |
| Cp1253          | 0x100204E5      |                                |
| Cp1254          | 0x100204E6      |                                |
| Cp1255          | 0x100204E7      |                                |
| Cp1256          | 0x100204E8      |                                |
| Cp1257          | 0x100204E9      |                                |
| Cp943C          | 0x100203AF      |                                |
| Cp943           | 0x100203AF      |                                |
| Cp949C          | 0x100203B5      |                                |
| Cp949           | 0x100203B5      |                                |
| Cp1363C         | 0x10020553      |                                |
| Cp1363          | 0x10020553      |                                |
| Cp950           | 0x100203B6      |                                |
| Cp1381          | 0x10020565      |                                |
| Cp1386          | 0x1002056A      |                                |
| EUC_JP          | 0x00030010      |                                |
| EUC_KR          | 0x0004000A      |                                |
| EUC_TW          | 0x00050010      |                                |
| Cp964           | 0x100203C4      |                                |
| Cp970           | 0x100203CA      |                                |
| Cp1383          | 0x10020567      |                                |
| Cp33722C        | 0x100283BA      |                                |
| Cp33722         | 0x100283BA      |                                |
| Cp930           | 0x100203A2      |                                |
| Cp1047          | 0x10020417      |                                |
| UCS2            | 0x00010100      | Valid only for ORBWCharDefault |
| UTF8            | 0x05010001      |                                |
| UTF16           | 0x00010109      | Valid only for ORBWCharDefault |

For more information, read the CORBA/IIOP specification, cited in "Resources for learning."

---

## Object Request Brokers: Resources for learning

Use the following links to find relevant supplemental information about Object Request Brokers (ORBs). The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.


These links are provided for convenience. Often, the information is not specific to this product but is useful all or in part for understanding the product. When

possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.



View links to additional information about:

- Planning, business scenarios, and IT architecture
- Administration
- Programming specifications


#### **Planning, business scenarios, and IT architecture**

-  **CORBA FAQ** (<http://www.omg.org/gettingstarted/corbafaq.htm>)  
Getting started with object request brokers and CORBA.

#### **Administration**

-  **IANA Character Set Registry**  
(<http://www.iana.org/assignments/character-sets>)  
This contains a list of all valid character encoding schemes.
-  **WebSphere Interoperability between Versions 3.5.x and 4.0.x**  
([http://www7b.boulder.ibm.com/wsdd/library/techarticles/0202\\_sundman/sundman.html](http://www7b.boulder.ibm.com/wsdd/library/techarticles/0202_sundman/sundman.html))  
This WebSphere Developer Domain article by Joel Sundman and Matt Kelm (February 2002, updated May 2002) is not directly related to the Java ORB service, but it touches upon ORB-related issues.

#### **Programming specifications**

-  **Catalog Of OMG CORBA/IIOP Specifications**  
([http://www.omg.org/technology/documents/corba\\_spec\\_catalog.htm](http://www.omg.org/technology/documents/corba_spec_catalog.htm))



---

## Chapter 4. Balancing workloads with clusters

To monitor application servers and manage the workloads of servers, use server clusters and cluster members provided by the Network Deployment product.

To assist you in understanding how to configure and use clusters for workload management, below is a scenario. In this scenario, client requests are distributed among the cluster members on a single machine. (A client refers to any servlet, Java application, or other program or component that connects the end user and the application server that is being accessed.) In more complex workload management scenarios, you can distribute cluster members to remote machines.

### Steps for this task

1. Decide which application server you want to cluster.
2. Decide whether you want to configure replication domains and entries. Replication enables the sharing of data among processes and the backing up of failed processes.
3. Deploy the application onto the application server.
4. After configuring the application server and the application components exactly as you want them to be, create a cluster. The original server instance becomes a cluster member that is administered through the cluster.
5. If you did not do so when creating a cluster, create one or more cluster members of the cluster.
6. Start all of the application servers by starting the cluster. Workload management automatically begins when you start the cluster members of the application server.
7. Stop the cluster.
8. Upgrade applications on clusters.
9. (Detect and handle problems with server clusters and their workloads.)

You need to define a bootstrap host for stand-alone Java clients, which are clients located on a different machine from the application server that have no administrative server. Add the following line to the Java Virtual Machine (JVM) arguments for the client:

```
-Dcom.ibm.CORBA.BootstrapHost=machine_name
```

where *machine\_name* is the name of the machine on which the administrative server is running.

---

## Workload management (WLM)

Workload management optimizes the distribution of client processing tasks. Incoming work requests are distributed to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests. Workload management also provides failover when servers are not available, improving application availability.

Workload management provides the following benefits to WebSphere Application Server applications:

- It balances client workloads, allowing processing tasks to be distributed according to the capacities of the different machines in the system.
- It provides failover capability by redirecting client requests if one or more servers is unable to process them. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With clustering, additional instances of servers, servlets, and other objects can easily be added to the configuration.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes the administration of servers and other objects.

In the WebSphere Application Server environment, you implement workload management by using clusters, transports, and replication domains.

## Techniques for managing state

Multimachine scaling techniques rely on using multiple copies of an application server; multiple consecutive requests from various clients can be serviced by different servers. If each client request is completely independent of every other client request, it does not matter whether consecutive requests are processed on the same server. However, in practice, client requests are not independent. A client often makes a request, waits for the result, then makes one or more subsequent requests that depend on the results received from the earlier requests. This sequence of operations on behalf of a client falls into two categories:

### Stateless

A server processes requests based solely on information provided with each request and does not rely on information from earlier requests. In other words, the server does not need to maintain state information between requests.

### Stateful

A server processes requests based on both the information provided with each request and information stored from earlier requests. In other words, the server needs to access and maintain state information generated during the processing of an earlier request.

For stateless interactions, it does not matter whether different requests are processed by different servers. However, for stateful interactions, the server that processes a request needs access to the state information necessary to service that request. Either the same server can process all requests that are associated with the same state information, or the state information can be shared by all servers that require it. In the latter case, accessing the shared state information from the same server minimizes the processing overhead associated with accessing the shared state information from multiple servers.

The load distribution facilities in WebSphere Application Server use several different techniques for maintaining state information between client requests:

- Session affinity, where the load distribution facility recognizes the existence of a client session and attempts to direct all requests within that session to the same server.
- Transaction affinity, where the load distribution facility recognizes the existence of a transaction and attempts to direct all requests within the scope of that transaction to the same server.

- Server affinity, where the load distribution facility recognizes that although multiple servers might be acceptable for a given client requests, a particular server is best suited for processing that request.

---

## Clusters

Clusters are sets of servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines, as opposed to the servers that are part of the same node and must be located on the same host machine.

A cell can have no clusters, one cluster, or multiple clusters.

Servers that belong to a cluster are *members* of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data. One cluster member might be running on a huge multi-processor enterprise server system while another member of that same cluster might be running on a small laptop. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical.

*A vertical cluster* has cluster members on the same node. *A horizontal cluster* has cluster members on multiple nodes.

A network dispatcher routes application access among cluster members by server-weighting, to provide better distribution control.

WebSphere Application Server can respond to increased use of an enterprise application by automatically replicating the application to additional cluster members as needed. This lets you deploy an application on a cluster instead of on a single node, without considering workload.

---

## Creating clusters

You can manage application servers collectively using a cluster. To create a cluster, view information about clusters, or manage server members on a cluster, use the Server Cluster page.

### Steps for this task

1. Go to the Server Cluster page. Click **Servers > Clusters** in the console navigation tree. The Server Cluster page lists clusters of application servers in the cell and states whether a cluster is stopped, started or unavailable.
2. Click **New** to access the Create New Cluster page.
3. Type a cluster name.
4. **(Optional)** To enable or disable node scoped routing optimization, place a checkmark in the **Prefer local enabled** check box. The default is enabled, which indicates that, if possible, EJB requests are routed to the client's node. If you enable this feature, performance is improved because client requests are sent to local EJBs.
5. **(Optional)** To enable memory-to-memory replication of HttpSession (for failover) or replication of cached data and cache invalidations with a Web Container's dynamic caching, select options supporting data replication.

6. Choose whether to create an empty cluster or to create a cluster based on an existing server.  
To create an empty cluster, do not include an existing server in this cluster.  
To create a cluster based on an existing server member, add server members to this cluster. To add a server member, choose **Select an existing server to add to this cluster** and then, from the drop-down list, select the server you want to add.
7. Click **Next**.
8. **(Optional)** Add application servers (cluster members) to the cluster. For each new cluster member, do the following:
  - a. Type the name of a new application server (cluster member) to add to the cluster.
  - b. Select the node on which the server will reside.
  - c. Specify the server weight. The weight value controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the servers' workload. The value can range from 0 to 20.
  - d. Specify whether to generate a unique HTTP port.
  - e. Specify whether to create a replication entry for the server. A replication entry enables memory-to-memory replication of HttpSession (for failover) or replication of cached data and cache invalidations with a Web Container's dynamic caching.
  - f. Specify the server template.
  - g. Click **Apply** to finish the cluster member. Repeat the above steps to define another cluster member.
9. Click **Next** and review the summary of changes.
10. Click **Finish** to complete the configuration.
11. Click **Save** on the administrative console taskbar and save your administrative configuration. As part of saving the change to the configuration, you can select **Synchronize changes with Nodes** before clicking **Save** on the Save page.
12. Before you can start the cluster, the configuration needs to be synchronized to the nodes. If you selected **Synchronize changes with Nodes** when saving your configuration in the previous step, you can ignore this step. If you are running automatic synchronization, wait until synchronization runs. Or, run manual synchronization to get the configuration files moved to the nodes. Click **System Administration > Nodes** and, on the Nodes page, select the node and click **Synchronize** or **Full Resynchronize**. The Nodes page displays status indicating whether the node is synchronized.
13. To further configure a cluster, click on the cluster's name under **Name**. This displays the settings for the server cluster instance. Note that, unless you have clicked **Save** and saved your administrative configuration, you only see the **Configuration** and **Local Topology** tabs; to see the **Runtime** tab as well you must save your administrative configuration. Also, ensure that changes are synchronized to the nodes (step 12).

## Server cluster collection

Use this page to view information about and manage clusters of application servers.

To view this administrative console page, click **Servers > Clusters**.

Click **New** to access the Create New Cluster page, which you use to define a new cluster.

### **Name**

Specifies a logical name for the cluster. The name must be unique among clusters within the containing cell.

### **Status**

Specifies whether cluster members are stopped, starting, or running.

If all cluster members are stopped, the cluster status and state is *Stopped*. After you request to start a cluster by clicking **Start** or **Ripplestart**, the cluster state briefly changes to *Starting* and each server that is a member of that cluster launches, if it is not already running. When the first member launches, the state changes to *PartialStart*. The state remains *PartialStart* until all cluster members are running, then the state changes to *Running* and the status is *Started*. Similarly, when stopping a cluster by clicking **Stop** or **ImmediateStop**, the state changes to *PartialStop* as the first member stops and changes to *Stopped* when all members are not running.

## **Server cluster settings**

Use this page to view or change the configuration and local topology of a server cluster instance. Provided you saved your administrative configuration after creating the server cluster instance, you can also view run-time information such as the status of the server cluster instance.

To view this administrative console page, click **Servers > Clusters > cluster\_name**.

### **Cluster name**

Specifies a logical name for the cluster. The name must be unique among clusters within the containing cell.

Data type String

### **Cluster short name**

Specifies the cluster short name for this cluster.

The name is 1-8 characters, alpha-numeric or national language. It cannot start with a numeric.

Data type String

### **Unique Id**

Specifies the unique ID of this cluster.

The unique ID property is read only. The system automatically generates the value.

### **Prefer local**

Specifies whether enterprise bean requests are routed to the node on which the client resides, if it is possible to do so.

Select the **Prefer Local** check box to specify routing of requests to the node on which the client resides. By default, the **Prefer Local** check box is selected, specifying routing of requests to the node.

Data type Boolean

Default true

### wlclD

Specifies the currently registered workload controller (WLC) identifier for the cluster. This setting might not display for all configurations.

Data type String

### State

Specifies whether cluster members are stopped, starting, or running.

If all cluster members are stopped, the cluster state is *websphere.cluster.stopped*. After you request to start a cluster, the cluster state briefly changes to *websphere.cluster.starting* and each server that is a member of that cluster launches, if it is not already running. When the first member launches, the state changes to *websphere.cluster.partial.start*. The state remains *websphere.cluster.partial.start* until all cluster members are running, then the state changes to *websphere.cluster.running*. Similarly, when stopping a cluster, the state changes to *websphere.cluster.partial.stop* as the first member stops and changes to *websphere.cluster.stopped* when all members are not running.

Data type String  
Range Valid values are *websphere.cluster.starting*, *websphere.cluster.partial.start*, *websphere.cluster.running*, *websphere.cluster.partial.stop*, or *websphere.cluster.stopped*.

---

## Creating cluster members

You create a cluster member to represent an application server in a cluster. To create a cluster member, view information about cluster members, or manage members of a cluster, use the Cluster Members page.

### Steps for this task

1. Go to the Cluster Members page. Click **Servers > Clusters** in the console navigation tree. Then, click a cluster in the collection of clusters and click **Cluster Members**. The Cluster Members page lists members of a cluster, states the nodes on which members reside, and states whether members are started, stopped or encountering problems.
2. Click **New** and follow the steps on the Create New Cluster Members page.
  - a. Type a name for the cluster member (application server).
  - b. Select the node on which the server will reside.
  - c. Specify the server weight. The weight value controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the servers' workload. The value can range from 0 to 100.
  - d. Specify whether to generate a unique HTTP port.
  - e. Specify whether to create a replication entry for the server.
  - f. Specify the server template.

- g. Click **Apply** to finish the cluster member. Repeat steps 1 through 7 to define another cluster member.
    - h. Click **Next**.
    - i. Review the summary of information on new cluster members and click **Finish**.
  3. Click **Save** on the administrative console taskbar and save your administrative configuration.
  4. To examine a cluster member's settings, click on the member's name under **Member Name** on the Cluster Members page. This displays the settings page for the cluster member instance.

---

## Cluster member collection

Use this page to view information about and manage members of an application server cluster.

To view this administrative console page, click **Servers > Clusters > *cluster\_name* > Cluster Members**.

### Member name

Specifies the name of the server in the cluster. On most platforms, the name of the server is the process name. The name must match the (object) name of the application server.

### Node

Specifies the name of the node for the cluster member.

### Status

Specifies whether a cluster member is running, stopped, or unavailable.

If a cluster member is stopped, its status is *Stopped*. After you request to start a cluster member by clicking **Start**, the status becomes *Started*. After you click **Stop**, its status changes to *Stopped* when it stops running.

Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the cluster member.

## Cluster member settings

Use this page to configure a member instance of an application server cluster.

To view this administrative console page, click **Servers > Clusters > *cluster\_name* > Cluster Members > *cluster\_member\_name***.

### Member Name

Specifies the name of the server in the cluster. On most platforms, the name of the server is the process name. The name must match the (object) name of the application server.

Data type

String

## Weight

Controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the server workload.

|           |         |
|-----------|---------|
| Data type | Integer |
| Range     | 0 to 20 |

## Unique ID

Specifies a numerical identifier for the application server that is unique within the cluster. The ID is used for affinity.

|           |         |
|-----------|---------|
| Data type | Integer |
|-----------|---------|

---

## Replication

WebSphere Application Server provides a service that transfers data or events among WebSphere Application Server servers. The service is called *WebSphere Internal Replication*, or *replication* for short.

The replication service transfers both J2EE application data and any internal data used to maintain the application data among WebSphere run-time processes in a cluster of application servers.

Currently, the Web container in WebSphere Application Server leverages replication.

The replication service can replicate HttpSession data among processes and retrieve the HttpSession if the process that currently maintains the HttpSession fails. Using replication for HttpSession failover provides a potentially lower cost and more easily administrable alternative to storing HttpSession in a relational database. Further, the service can distribute across a WebSphere cluster information on invalid data and actual cached data maintained by a Web container's dynamic caching.

## Replication entry

A replication entry (or *replicator*) is a run-time component that handles the transfer of internal WebSphere Application Server data.

WebSphere Application Server processes can connect to any replicator within a domain to receive data from other processes connected to any other replicator in the same domain. If the replicator a process is connected to goes down, the WebSphere Application Server process automatically attempts to reconnect to another replicator in the domain and recover data missed while unconnected.

You can define replicators to operate within a running application server process. Replicators are not enabled by default. You must define replicators as needed as part of application server and cluster management.

You can take the default settings for replicators or specify settings values for replicators that better suit your server configuration. The default configuration options are suitable for many scenarios.



## Replication domain

A replication domain is a collection of replicator entry (or *replicator*) instances used by clusters or individual servers within a cell.

All replicators within a replication domain connect with each other, forming a network of replicators.

The default is to define a replication domain for a cluster when creating the cluster. However, replication domains can span across clusters.

Global default settings apply to all replication use for a given replication domain across a cell. Most default settings tune and control the behavior of replicator entries in managed servers across the cell. Such default settings control the use of encryption or the serialization and transferring of objects. Some default settings tune and control how specific WebSphere Application Server functions (for example, session manager and dynamic caching) leverage replication, such as session use of partitions.

For situations that require settings values other than the default, change the values for a given replication domain on the Internal Replication Domains page. Settings include various resource allocation, replication strategies (such as grouping or partitioning) and methods, as well as some security related items.

If you are using replication for HttpSession failover, you might need to filter where the session replicates to. For example, only replicate to two places out of many. The global default settings define the partition size or number of groups and the session manager settings define the groups to which a particular instance belongs.

Filtering is less important if you are using replication to distribute information on invalid data and actual cached data maintained by a Web container's dynamic caching. Replication does not occur for failover as much as for data synchronization across a cluster or cell when you likely want to avoid expensive costs for generating data potentially needed across those various servers.

Note that you can filter or segment by using multiple replication domains.

---

## Replicating data

To enable the sharing of data among processes and the backing up of failed processes, you can use the replication service provided by WebSphere Application Server. To use the service, you define replication domains, which list interconnected replicator entries (residing in managed servers in the cell) that can exchange data.

There are two ways to define replication domains and replicator entries:

- You can use the Internal Replication Domains page and Replicator Entry page to define replication domains and replicator entries. To access the Internal Replication Domains page, click **Environment > Internal Replication Domains** in the console navigation tree. To access the Replicator Entry page, click a replication domain on the Internal Replication Domains page and then click **Replicator Entries**. When you create the entries on the Replicator Entry page, you can select any server for the replicator to reside in. The page lists all servers in the cell that do not already have replicators defined.
- You can define replication domains and replicator entries when you create a cluster on the Create New Cluster page. Using the page allows you to create a

replication domain that has the same name as the cluster and, as you add or create new application servers in the cluster, define replicator entries in those servers. To access the Create New Cluster page, click **Servers > Clusters** in the console navigation tree to go to the Server Clusters page and click **New**.

A replicator does not need to run in the same process as the application server that uses it. However, it might be easier to manage replicators and replication domains if a one-to-one correspondence exists between replicators and application servers. During configuration, an application server connects by default to its local replicator, so you do not need to explicitly specify the replicator to use. All processes share equally in the replication cost.

#### Steps for this task

1. (Create an application server). Later, enable a replication domain and its replicators (step 2).

Or, create a cluster and add an application server to it. When you define the cluster, you can specify that you want a replication domain associated with the cluster. Also, when you define a cluster, you can specify that you want a replicator associated with an application server. For example, you might specify that a replicator launch in the same Java virtual machine as a Web container. Or, you can enable a replicator later (step 2).
2. Create a replication domain if one is not already created for the processes you want supported by data replication. Go to the Replication Domains page and click **New**. On the settings for a replication domain instance, specify values for the instance. The default values generally will be sufficient, especially as to pooling and timeout values.
  - a. Name the replication domain.
  - b. Specify the timeout interval.
  - c. Specify the encryption type. The DES and TRIPLE\_DES options encrypt data sent between WebSphere Application Server processes and better secure the network joining the processes.
  - d. Partition the replication domain to filter the number of processes to which data is sent. Partitioning the replication domain is most often done if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. Partitioning is not supported for sharing of cached data maintained by Web container dynamic caching.
  - e. Specify whether you want a single replication of data to be made. Enable the option if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails.
  - f. Specify whether processes should receive data in objects or bytes. Processes receiving data in objects receive the data and class definitions. Processes receiving data in bytes receive the data only.
  - g. **(Optional)** Configure a pool of replication resources. Pooling replication resources can enhance the performance of the internal data replication service.
3. Create replicators for the processes you want supported by data replication, if replicators have not already been created for the processes. The default convention is to define a replicator in each application server that uses replication. However, you can define a pool of replicators, separate from the servers hosting applications.
  - a. Click on the replication domain instance on the Replication Domains page and then **Replicator Entries** to access the Replicator Entry page.

- b. Click **New** and, on the replicator entry settings page, define a replicator. Specify a replicator name and, from the drop-down list of the available servers within the cell to which you can assign a replicator, select a server. Also specify a host name and ports. Note that a replicator has two end points (replicator and client end points) that use the same host name but have different ports.
4. If you use the DES or TRIPLE\_DES encryption type for a replicator, click **RegenerateKey** on the settings for a replication domain instance at regular intervals, such as monthly.  
Periodically changing the key enhances security.

---

## Internal replication domain collection

Use this page to view and manage replicator instances used within a cell. Replicators can transfer both application data and any internal data used to maintain the application data among WebSphere Application Server run-time processes in a cluster of application servers.

To view this administrative console page, click **Environment > Internal Replication Domains**.

Using replicators, you can replicate HttpSession data among processes and retrieve the HttpSession if the process that currently maintains the HttpSession fails. Further, you can distribute across a cell the creation, modification, and invalidation of cached data maintained by a Web container's dynamic caching.

If you are using replication for HttpSession failover, you will likely need to filter where the session replicates to. For example, only replicate to two places out of many. The global default settings define the partition size or number of groups and the session manager settings define the groups to which a particular instance belongs. Filtering is less important if you are using replication to distribute information on cached data maintained by a Web container's dynamic caching.

The default is to define a replication domain for a cluster when creating the cluster. However, you can create a new domain from this page. Click **New** and follow the instructions on the page displayed.

Clicking **Delete** deletes a domain and all replicators defined under the domain.

### Name

Specifies a name for the replication domain.

---

## Internal replication domain settings

Use this page to configure a replicator instance.

To view this administrative console page, click **Environment > Internal Replication Domains > *replication\_domain\_name***.

An application server connected to replicator within a domain can access the same set of data sent out by any application server connected to any other replicator (including the same replicator). Data is not shared across replicator domains.

## Name

Specifies a name for the replication domain.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Request Timeout

Specifies the number of seconds that a replicator waits when requesting information from another replicator before giving up and assuming the information does not exist. The default is 5 seconds.

|           |         |
|-----------|---------|
| Data type | Integer |
| Units     | Seconds |
| Default   | 5       |

## Encryption Type

Specifies the type of encryption used before transfer. The options include NONE, DES, TRIPLE\_DES. The default is NONE. The DES and TRIPLE\_DES options encrypt data sent between WebSphere processes and better secure the network joining the processes.

If you specify DES or TRIPLE\_DES, a key for global data replication is generated after you click **Apply** or **OK**. When you use the DES or TRIPLE\_DES encryption type, click **RegenerateKey** at regular intervals such as monthly because periodically changing the key enhances security.

|           |        |
|-----------|--------|
| Data type | String |
| Default   | NONE   |

## DRS Partition Size

Specifies the number of groups into which a replication domain is partitioned. By default, data sent by a WebSphere Application Server process to a replication domain is transferred to all other WebSphere Application Server processes connected to that replication domain. To filter or reduce the number of destinations for the data being sent, partition the replication domain. The default partition size is 10, and the partition size should be 10 or more to enhance performance.

Partitioning the replication domain is most often done if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. Partitioning is not supported for sharing of cached data maintained by Web container dynamic caching. As to dynamic caching, all partitions or groups are always active and used for data replication.

When you partition a replication domain, you define the total number of groups or partitions. Use this setting to define the number of groups. Then, when you configure a specific session manager under a Web container or as part of an enterprise application or Web module, select the partition to which that session manager instance listens and from which it accepts data. To specify the groups to which an application server listens, change the settings for affected servers on a Session Manager page. In addition, you can set a *replicator role* for a server. This replicator role affects whether a WebSphere process sends data to the replication domain, receives data, or does both. The default is both to receive and send data.

|           |         |
|-----------|---------|
| Data type | Integer |
| Default   | 10      |

## Single Replica

Specifies that a single replication of data be made. Enable this option if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. This option restricts the recipient of the data to a single instance.

This setting provides filtering beyond grouping or partitioning. Using this setting, you can choose to have data only sent to one other listening instance in the replication domain.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

## Serialization Method

Specifies the object serialization method to use when replicating data. An administrative concern with replicating Java objects is locating the class definition, especially in a J2EE environment where class definitions might reside only in certain web modules or enterprise applications. Object serialization methods define whether the processes receiving data also need the class definition.

The options for this setting are OBJECT and BYTES. The default is BYTES.

OBJECT instructs a replicator to write the object directly to the stream. With OBJECT, a replicator must reinstantiate the object on the receiving side so must have the class definition.

BYTES instructs a replicator to break down the object into bytes and then send only the bytes across the stream. With BYTES, a replicator does not need to instantiate the object on the receiving side. The BYTES option is useful for failover, where the data is not used at the receiving side and thus the class definitions do not need to be stored there. Or, the option requires that you move class definitions from the Web application class path to the system class path.

|           |                                   |
|-----------|-----------------------------------|
| Data type | String                            |
| Default   | BYTES                             |
| Range     | Valid values are OBJECT or BYTES. |

## DRS Pool Size

Specifies the maximum number of items allowed in a pool of replication resources. The default is 10.

Pooling replication resources can enhance the performance of the WebSphere internal data replication service.

|           |         |
|-----------|---------|
| Data type | Integer |
| Default   | 10      |
| Range     | 1 to 50 |

## DRS Pool Connections

Specifies whether the data replication service includes replicator connections in a pool of replication resources. Whether this option is enabled or not, the pool includes replicator sessions, publishers and subscribers.

The default is not to include replicator connections in the pool.

|           |         |
|-----------|---------|
| Data type | Boolean |
| Default   | false   |

---

## Replicator entry collection

Use this page to view and manage replicator entries.

To view this administrative console page, click **Environment > Internal Replication Domains > *replication\_domain\_name* > Replicator Entries**.

To configure a new replicator entry, click **New** and follow the instructions on the page. You add a replicator to an existing server in the cell.

## Replicator Name

Specifies a name for the replicator entry.

---

## Replicator entry settings

Use this page to view and configure a replicator entry (or *replicator*).

To view this administrative console page, click **Environment > Internal Replication Domains > *replication\_domain\_name* > Replicator Entries > *replicator\_entry\_name***.

Replicators communicate using TCP/IP. Thus, you must allocate an IP address and ports for replicators. Use this page to name a replicator and then to allocate an IP address and two ports (replicator and client ports) for the replicator.

## Replicator Name

Specifies a name for the replicator entry.

|           |        |
|-----------|--------|
| Data type | String |
|-----------|--------|

## Server

Specifies the server for which you are defining a replicator. The drop-down list provides the names of servers that do not already have replicators.

|           |        |
|-----------|--------|
| Data type | String |
| Default   | None   |

## Replicator and Client Host Name

Specifies the IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP file, or HTML page).

A replicator port and client port share the same host name.

|           |        |
|-----------|--------|
| Data type | String |
| Default   | None   |

## Replicator Port

Specifies the port for which the Web server is configured to accept client requests. The port value is used in conjunction with the host name.

The replicator port enables communication among replicators. It provides replicator port to replicator communication. The usual value specified is 7874.

|           |         |
|-----------|---------|
| Data type | Integer |
| Default   | None    |

## Client Port

Specifies the port for which the Web server is configured to accept client requests. The port value is used in conjunction with the host name.

The client port enables communication between an application server process and a replicator. It provides client port to application server communication. The usual value specified is 7873.

|           |         |
|-----------|---------|
| Data type | Integer |
| Default   | None    |

---

## Starting clusters

You can start all members of a cluster at the same time by requesting that the state of a cluster change to *running*. That is, you can start all application servers in a server cluster at the same time.

When you request that all members of a cluster start, the cluster state changes to *websphere.cluster.partial.start* and each server that is a member of that cluster launches, if it is not already running. After all members of the cluster are running, the cluster state becomes *websphere.cluster.running*.

### Steps for this task

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.
2. Put a checkmark in the check boxes beside those clusters whose members you want started.
3. Click **Start** or **RippleStart**.
  - **Start** launches the server process of each member of the cluster by calling the node agent for each server to start the servers. After all servers are running, the state of the cluster changes to *websphere.cluster.running*. If the call to a node agent for a server fails, the server will not start.
  - **RippleStart** combines stopping and starting operations. It first stops and then restarts each member of the cluster.

---

## Stopping clusters

You can stop all members of a cluster at the same time by requesting that the state of a cluster change to *stopped*. That is, you can stop all application servers in a server cluster at the same time.

### Steps for this task

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.
2. Put a checkmark in the check boxes beside those clusters whose members you want stopped.
3. Click **Stop** or **Immediate Stop**.
  - **Stop** halts each server in a manner that allows the server to finish existing requests and allows failover to another member of the cluster. When the stop operation begins the cluster state changes to *websphere.cluster.partial.stop*. After all servers stop, the cluster state becomes *websphere.cluster.stopped*.
  - **Immediate Stop** brings down the server quickly without regard to existing requests. When the stop operation begins, the cluster state changes to *websphere.cluster.partial.stop*. After all servers stop, the cluster state becomes *websphere.cluster.stopped*.

You can also stop and start server clusters from the settings page for a server cluster instance. To access such a page, click on the server cluster that you want to start or stop in the collection under **Name** on a Server Cluster page. You can view the status of a server cluster (that is, whether the cluster is started or stopped) on the **Runtime** tab of the settings page for a server cluster instance. Note that the **Runtime** tab is only shown if you have clicked **Save** on the administrative console taskbar since creating the server cluster instance.

**Note to Windows users:** If you start and stop application servers that are part of a cluster using the Windows Services facility, the cluster state does not always update correctly. For example, if a cluster is running and you stop a cluster member through the Services GUI, the cluster state remains as *Started* even though the server is no longer running.

---

## Tuning a workload management configuration

You can set values for several workload management client properties to tune the behavior of the workload management run time. You set the properties as command-line arguments for the Java virtual machine (JVM) process in which the workload management client is running.

**Caution:** Set the values of these properties only in response to problems that you encounter. In most cases, you do not need to change the values. If workload management is functioning correctly, changing the values can produce undesirable results.

To change the property values, you can use the Java Virtual Machine page of the administrative console or use the wsadmin tool. In cases such as where a servlet is a client to an enterprise bean, use the administrative console page for the application server where the servlet is running to configure the properties. The steps below describe how to change the values using the console.

### Steps for this task



1. Access the (Java Virtual Machine page).
  - a. Click **Servers > Application Servers** in the console navigation tree.
  - b. On the (Application Server page), click on the name of the server where the client is running.
  - c. On the (settings page for the selected application server), click **Process Definition**.
  - d. On the (Process Definition page), click **Java Virtual Machine**.
2. On the (Java Virtual Machine page), specify one or more of the following command-line arguments in the **Generic JVM arguments** field:

**-Dcom.ibm.CORBA.RequestTimeout=<i>timeout\_interval</i>**

If your application is experiencing problems with timeouts, this argument changes the value for the com.ibm.CORBA.RequestTimeout property, which specifies the timeout period for responding to requests sent from the client. This argument uses the -D option. *timeout\_interval* is the timeout period in seconds. If your network experiences extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that participates in workload management can time out before it receives a response.

**Note:** Be careful specifying this property; it has no recommended value. Set it only if your application is experiencing problems with timeouts.

**-Dcom.ibm.websphere.wlm.unusable.interval=<i>interval</i>**

If the workload management state of the client is refreshing too soon or too late, this argument changes the value for the com.ibm.websphere.wlm.unusable.interval property, which specifies the time interval that the workload management client run time waits after it marks a server as unavailable before it attempts to contact the server again. This argument uses the -D option. *interval* is the time in seconds between attempts. The default value is 300 seconds. If the property is set to a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after the time period has ended.

3. Click **OK**.
4. (Stop the application server) and then (restart the application server).

---

## Workload management run-time exceptions

The workload management service can throw the following exceptions if it encounters problems:

**org.omg.CORBA.TRANSIENT with a minor code 1229066306 (0x40421042)**

This exception is thrown if the workload management routing service cannot retry a request and the failure resulted from a connection error. This exception indicates that the application should invoke some compensation logic and resubmit the request.

**org.omg.CORBA.NO\_IMPLEMENT with a minor code 1229066304 (0x49421040)**

This exception is thrown if the workload management service cannot contact any of the EJB application servers that participate in workload management.

The WebSphere Application Server client can catch these exceptions and then implement its own strategies to handle the situation. For example, it can display an error message if no servers are available.

The workload management routing service can reroute a failed request to a different target transparently to the application if the application will not be adversely affected by a second attempt. Currently, the only way is to check if the request did not execute in whole or part on the previous attempt. When a request executes in whole or in part, an *org.omg.CORBA.TRANSIENT with the minor code 1229066306 (0x49421042)* exception is thrown to signal that a request can be made again. This informs the application that another target might be available to satisfy the request, but the request could not be failed over transparently to the application. Thus, the application can resubmit the request. The routing service throws an *org.omg.CORBA.NO\_IMPLEMENT with the minor code 1229066304 (0x49421040)* exception if it cannot locate a suitable target for the request. The exception is thrown, for example, if the cluster is stopped or if the application does not have a path to any of the cluster members.

---

## Clustering and workload management: Resources for learning




Use the following links to find relevant supplemental information about clustering and workload management. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

View links to additional information about:

- Programming model and decisions
- Programming instructions and examples

### Programming model and decisions

-  **Improving availability by clustering the WebSphere Application Server** (<http://www-106.ibm.com/developerworks/ibm/library/i-extreme18/?open&l=937,t=gr>)
-  **Redbook on WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition** (<http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246153.html?Open>)
-  **Failover and Recovery in WebSphere Application Server Advanced Edition 4.0** (<http://www7.software.ibm.com/vadd-bin/ftpd!1/vadc/wsdd/pdf/modjeski.pdf>)

### Programming instructions and examples

-  **WebSphere Application Server education** (<http://www.ibm.com/software/webservers/learn/>)
-  **Listing of all IBM WebSphere Application Server Redbooks** (<http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere>)
-  **Redbook on IBM WebSphere V4.0 Advanced Edition Scalability and Availability** (<http://publib-b.boulder.ibm.com/Redbooks.nsf/9445fa5b416f6e32852569ae006bb65f/ff31072025dcf5de85256aca00781918?OpenDocument&Highlight=0,plug-in>)

---

## Chapter 5. Web services gateway: Enabling Web services

You use the IBM Web services gateway to handle Web service invocations between Internet and Intranet environments. You use it to make your internal Web services available externally, and to make external Web services available to your internal systems.

With the Web services gateway you can administer Web services, Channels, Filters and UDDI references.

Detailed instructions on how to enable Web services through the IBM Web services gateway are given in the following tasks:

- Web services gateway - Completing the installation.
- Administering the Web services gateway.
- Running the Web services gateway samples.
- Administering security for the Web services gateway.
- Troubleshooting the Web services gateway.

For a brief overview of what the Web services gateway is for, and how it works, see "Web services gateway - Frequently Asked Questions".

For a list of the major changes since the AlphaWorks preview version of the Web services gateway, see "Web services gateway - What is new in this release".

For more information about working with Web services, visit the Internet sites referenced in Web services gateway: Resources for Learning.

---

### Web services gateway - Frequently Asked Questions

This topic provides answers to the following set of frequently asked questions about the Web services gateway.

#### What are Web services?

Web services are modular applications that interact with one another across the Internet. Web services are based on shared, open and emerging technology standards and protocols (such as SOAP, UDDI, and WSDL) and can communicate, interact, and integrate with other applications, no matter how they are implemented.

#### What is the IBM Web services gateway?

The gateway is a middleware component that bridges the gap between Internet and Intranet environments during Web service invocations. You use it to manage

- Web services.
- channels that carry requests to and responses from the services.
- filters that act upon the services.
- references to UDDI registries in which services can be registered.

#### How does the Web services gateway work?

The gateway builds upon the Web Services Description Language (WSDL)(<http://www.w3.org/TR/wsdl>) and the Web Services Invocation Framework (WSIF) for deployment and invocation.

You deploy a Web service to the Web services gateway by deploying a WSDL file that describes how the Web services gateway should access it. The WSDL file can be deployed to a UDDI registry or to a URL. You can send requests passing through the Web services gateway to a Java class, an enterprise bean, a SOAP server or a SOAP/JMS server (including another gateway).

A request to the Web services gateway arrives through a channel, is translated into an internal form, then passed through any filters that are registered for the requested service, and finally sent on to the service implementation. Responses follow the same path in reverse.

#### **What problems are solved by the Web services gateway?**

- **Securely "externalizing" Web services:** Business applications that are exposed as Web services can be used by any Web service-enabled tool, regardless of the implementation details, to create new applications. To better integrate your business processes, you might want to expose these assets to business partners, customers and suppliers who are outside the firewall. The Web services gateway lets clients from outside the firewall use Web services that are buried deep inside your enterprise. The gateway also allows you to set access control on each of these deeply-buried services.
- **Better return on investment:** A process that you develop as a Web service can be reused by any number of partners.
- **Use of existing infrastructure:** With the Web services gateway, you can use your existing messaging infrastructure to make Web service requests, and use your existing Web services for external process integration.
- **Protocol transformation:** You might use one particular messaging protocol to invoke Web services, while your partners use some other protocol. Using the Web services gateway, you can trap the request from the client and transform it to another messaging protocol.

#### **Who should use the Web services gateway?**

Any enterprise that chooses to share its resources selectively with its business partners and customers. IT managers and developers, who deploy resources, can also benefit from this technology.

---

## **Web services gateway - What is new in this release**

The Web services gateway was first made available on AlphaWorks(<http://www.alphaworks.ibm.com/tech/wsgw>) on 21 December 2001. The main differences between the AlphaWorks edition and this version are as follows:

- The gateway has been rebuilt using enterprise beans.  
**Note:** A side-effect of this is that the Web services gateway now only runs in an application server that has an EJB container. So it no longer runs in the Tomcat server.
- The gateway includes UDDI integration, so you can deploy and remove Web services to a UDDI registry as well as to a URL.
- The gateway supports bidirectional interactions (that is, both inbound and outbound requests) directly, by deploying two instances of each type of channel.

**Note:** To achieve this configuration with the AlphaWorks version, you had to deploy two instances of the Web services gateway; one for inbound communication and one for outbound communication.

- *Interceptors* have been renamed as *filters*.
- Channels, filters and UDDI references are deployed to the Web services gateway, then associated with individual Web services. So when you configure a Web service, you choose the following entities:
  - The channels on which it is available.
  - The filters (if any) which apply to it.
  - The UDDI references (if any) to which it is deployed.
- You can change the channels, filters and UDDI references that are associated with a deployed service without having to remove the service.
- You can deploy multiple targets for a single service (that is, more than one implementation of a service that has the same service interface).
- You can set security (basic authorization) on the individual methods of a Web service, and for the gateway as a whole.
- The gateway can invoke Web services over HTTPS.

## Web services gateway - Completing the installation

### Before you begin

This task assumes that, when you [ ], you either chose to install the Web services gateway (by choosing the "custom install" option **Web services -> Web services gateway**) or you accepted the "typical install" option (which includes the gateway). If you did this, then all the files that are needed to run a Web services gateway were copied into directories under `<i>WebSphere_DeployMgr_root</i>`, where *WebSphere\_DeployMgr\_root* is the deployment manager root directory (by default WebSphere/DeploymentManager).

The following table lists the Web services gateway files, and the locations into which they are placed by the install. The **Location** column shows the subdirectory under *WebSphere\_DeployMgr\_root*. For example, if you installed WebSphere Application Server onto a machine running Windows, and accepted the default directory names, then the location of the "installableApps" directory is `C:\Program Files\WebSphere\DeploymentManager\installableApps`.

| File name               | Purpose                                                           | Location              |
|-------------------------|-------------------------------------------------------------------|-----------------------|
| wsgw.ear                | The Web services gateway application                              | /installableApps      |
| wsgwsoap1.ear           | The Apache SOAP channel application number 1                      | /installableApps      |
| wsgwsoap2.ear           | The Apache SOAP channel application number 2                      | /installableApps      |
| wsgwcorr.ear            | The application used to correlate any asynchronous reply messages | /installableApps      |
| wsgwauth.ear            | The Web service operation-level security application              | /installableApps      |
| WSGWResourceBundles.jar | System messages for the Web services gateway application          | /lib                  |
| Various install scripts | Installation of the Web services gateway                          | /WSGW/scripts/install |

|                           |                                                                                                                                         |                    |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| Authorization scripts     | Generation of authorization beans for Web service operation-level security                                                              | /WSGW/scripts/auth |
| Cloudscape directory tree | Cloudscape database containing the Web services gateway tables and pre-loaded data needed to install a database to use with the gateway | /bin/WSGWDB        |

To complete the gateway installation, you have two choices:

- Install into a deployment manager cell.
- Install into a stand-alone application server.

For production use you will probably want to install the gateway into one or more nodes of a deployment manager cell. But for development or test purposes you might find it useful to install the gateway into a stand-alone application server (that is, an application server that is not part of a deployment manager cell).

#### What to do next

To finish the Web services gateway installation, complete the following steps:

1. Confirm that your system configuration complies with the Web services gateway prerequisites and constraints.
2. Establish requirements for using a database with the gateway.
3. Enable security if required.
4. Either Install the gateway into a deployment manager cell, or Install the gateway into a stand-alone application server.
5. Test the installation.

---

## Web services gateway - prerequisites and constraints

The gateway is an enterprise application that you can deploy

- to an application server that is a node of a WebSphere Application Server Network Deployment cell.
- to a stand-alone application server.

WebSphere Application Server Network Deployment is not a stand-alone product for running enterprise applications. So if you want to deploy the gateway you also need to install the base WebSphere Application Server product.

**Note:** Although it is not installed by default, a copy of the base WebSphere Application Server product is packaged with the WebSphere Application Server Network Deployment product.

This version of the Web services gateway is also subject to the following constraints:

- WSDL definitions for target services must use XML Schema version 2001. For more information, see "Web services gateway troubleshooting tips".
- The gateway application (wsgw.ear) must be installed before channel and filter applications. If the gateway application needs to be reinstalled, all channels and filters must be uninstalled first, then reinstalled after the gateway application.

You might also find it useful to enable trace for all gateway components: Set the WebSphere Application Server trace string to `com.ibm.wsgw.*=all=enabled`, and have trace, stdout and stderr for the application server written to a well-known location. For information on how to do this, see [Enabling trace](#).

---

## Establishing requirements for using a database with the gateway

Before installing the Web services gateway you need to decide on your database requirements. There are three choices:

### No database

You do not have to install a database, but if you do not install one you cannot use asynchronous channels.

**DB2** If DB2 is already installed on your system, you can create an associated DB2 Web services gateway database.

### Cloudscape

You can use the copy of Cloudscape runtime that is installed with WebSphere Application Server.

### What to do next

Database installation instructions are included in the next task. So after you have decided on your database requirements you are ready to *either* Install the gateway into a deployment manager cell *or* Install the gateway into a stand-alone application server.

---

## Installing the gateway into a deployment manager cell

### Before you begin

This task assumes that you have already completed the step [Establishing requirements for using a database with the gateway](#).

If you want to enable gateway-level security, you must do so before you install the Web services gateway.

If you want to use DB2 on a UNIX system, the shell in which the `install` is invoked requires the environment to be set up using the `db2profile` and `usejdbc2` script files. To do this, run both the following commands:

```
./home/db2_instance/sql1lib/db2profile
./home/db2_instance/sql1lib/java12/usejdbc2
```

where `db2_instance` is the db2 instance to be configured.

In this task you install the gateway into an application server that is hosted on a node of an existing deployment manager cell. The major elements of this process are as follows:

- Database and Table creation (optional - DB2 only).
- JDBC driver and Datasource creation (optional - only if a database is being installed).
- Installation of the following enterprise applications:
  - The gateway application.
  - Apache SOAP channel 1.
  - The correlation application (optional - only if a database is being installed).

- Installation of other gateway applications. For example:
  - Apache SOAP channel 2.

To install the gateway into an application server that is hosted on a node of a deployment manager cell, complete the following steps:

#### Steps for this task

1. **(Optional)** To create and install a DB2 database and associated tables, complete the following steps:
  - a. From a command prompt, go to directory `<i>WebSphere_DeployMgr_root</i>/WSGW/scripts/install`, where `WebSphere_DeployMgr_root` is the deployment manager root directory (by default `WebSphere/DeploymentManager`).
  - b. Enter the command `WSGWinstallDB`:
    - **[Windows]** `WSGWinstallDB.bat <i>WebSphere_DeployMgr_root</i> <i>db2user_id</i> <i>db2password</i>`
    - **[UNIX]** `WSGWinstallDB.sh <i>WebSphere_DeployMgr_root</i> <i>db2user_id</i>`

For example (UNIX systems):

```
./WSGWinstallDB.sh /opt/WebSphere/DeploymentManager mydb2id
```

**Note:** Running `WSGWinstallDB` also creates `WSGWinstallDB.log` in the application server for network deployment's `/logs` directory. Open this file to check that the database was created successfully.

2. Start the application server (you can use the command `startServer<i>.ext</i> <i>your_server</i>`).
3. From a command prompt, go to directory `<i>WebSphere_DeployMgr_root</i>/WSGW/scripts/install`.
4. Clear your class path.  
You can use the following command:
  - (Windows systems): `set CLASSPATH=`
  - (UNIX systems): `unset CLASSPATH`
5. Enter the command `<i>WebSphere_DeployMgr_root</i>/bin/wsadmin<i>.ext</i> -f setupWSGW.jac1 <i>parm1 ... parmN</i>` where
  - `parm1` is the `<i>WebSphere_DeployMgr_root</i>` directory
  - `parm2` is the server name
  - `parm3` is the node name (this is case sensitive)

**If you are not using a database**, there are no more parameters and `wsgwcorr.ear` is not installed. For example (Windows systems):

```
C:\Progra~1\WebSphere\DeploymentManager\bin\wsadmin.bat -f
setupWSGW.jac1 C:/Progra~1/WebSphere/DeploymentManager server1 PHJ2
```

**Note:** The use of forward slashes (/) is compulsory for this command, even on Windows systems.

**If you are using Cloudscape**, there is one more parameter:



- *parm4* is the name and location of the WSGWDB directory (*<i>WebSphere\_DeployMgr\_root</i>/bin/WSGWDB*).

For example (Windows systems):

```
C:\Progra~1\WebSphere\DeploymentManager\bin\wsadmin.bat -f
setupWSGW.jacl C:/Progra~1/WebSphere/DeploymentManager server1 PHJ2
C:/Progra~1/WebSphere/DeploymentManager/bin/WSGWDB
```

**Note:** The use of forward slashes (/) is compulsory for this command, even on Windows systems.

**If you are using DB2,** there are four more parameters:

- *parm4* is WSGWDB.
- *parm5* is your DB2 user ID
- *parm6* is your DB2 password
- *parm7* is the name and location of file db2java.zip

For example (Windows systems):

```
C:\Progra~1\WebSphere\DeploymentManager\bin\wsadmin.bat -f
setupWSGW.jacl C:/Progra~1/WebSphere/DeploymentManager server1 PHJ2
WSGWDB mydb2id mydb2pw C:/Progra~1/SQLLIB/java/db2java.zip
```

**Note:** The use of forward slashes (/) is compulsory for this command, even on Windows systems.

When you run the setupWSGW.jacl command, the following initial set of gateway applications is installed:

- The gateway itself (wsgw.ear).
  - Apache SOAP channel 1 (wsgwsoap1.ear).
  - (Optionally) The correlation application (wsgwcorr.ear).
6. Install additional gateway applications. For example:
- Apache SOAP channel 2 (wsgwsoap2.ear)

To install additional gateway applications, complete the following steps:

**Note:** If you prefer, you can install these EAR files using the WebSphere Application Server administrative console, as described in the final step of the task Enabling operation-level authorization.

- From a command prompt, go to directory *<i>WebSphere\_DeployMgr\_root</i>/bin*
- To start the application server, enter the command *startServer<i>.ext</i><i>your\_server</i>*

where

- *.ext* is *.bat* for a Windows system and *.sh* for a UNIX system.
- *your\_server* is your application server's name.

For example (UNIX systems): *./startServer.sh server1*.

- To start the WebSphere administration program, enter the command *wsadmin<i>.ext</i>*.

- d. For each additional Web services gateway enterprise application that you want to install, enter the following commands at the wsadmin> prompt:
- ```
$AdminApp install path_to_ear_file {-appName application -server your_server  
-node your_node_name}
```

```
$AdminConfig save
```

where

- *application* is the name of the enterprise application
- *path_to_ear_file* is the name and location of the enterprise application's EAR file
- *your_node_name* is the node name (this is case sensitive)

For example (Windows systems):

```
$AdminApp install C:/Progra~1/WebSphere/DeploymentManager/installableApps  
/wsgwsoap2.ear {-appName wsgwsoap2 -server server1 -node PHJ2}  
$AdminConfig save
```

- e. After you have installed all your additional Web services gateway enterprise applications, close the WebSphere administration program by entering quit or exit at the wsadmin> prompt.
7. To stop then restart the application server, complete the following steps:
- a. Enter the command stopServer<i>.ext</i> <i>your_server</i>
 - b. Enter the command startServer<i>.ext</i> <i>your_server</i>

What to do next

If you want more than one gateway installation (for example to create a gateway cluster) then repeat the steps given in this topic for another application server that is hosted on a node in a deployment manager cell.

You are now ready to test the installation. Run the test on every application server on which you have installed the gateway.

Installing the gateway into a stand-alone application server

Before you begin

This task assumes that you have already completed the step Establishing requirements for using a database with the gateway.

If you want to enable gateway-level security, you must do so before you install the gateway.

If you want to use DB2 on a UNIX system, the shell in which the install is invoked requires the environment to be set up using the db2profile and usejdb2 script files. To do this, run both the following commands:

```
./home/db2_instance/sql1lib/db2profile  
./home/db2_instance/sql1lib/java12/usejdb2
```

where *db2_instance* is the db2 instance to be configured.

Note: The application server in which you run the gateway must not form part of a cell managed by a deployment manager. In other words, you must not issue an addNode command for the node containing the application server in which you run the Web services gateway application. If you do issue the addNode command, then

the installed Web services gateway application is deleted by the node synchronization process that takes place within a cell of application servers.

When you [], all the files that are needed to run a Web services gateway were copied into directories under *WebSphere_DeployMgr_root*, where *WebSphere_DeployMgr_root* is the deployment manager root directory (by default WebSphere/DeploymentManager). Before you can install and run the gateway in a single application server instance in your network space, you must first copy these files over to the application server. You can do this by completing the following steps:

- Stop the application server into which you plan to install the Web services gateway. You can use the command `stopServer<i>.ext</i> <i>your_server</i>` where
 - *.ext* is *.bat* for a Windows system and *.sh* for a UNIX system.
 - *your_server* is your application server's name.

For example (UNIX systems): `./stopServer.sh server1`.

- Copy all the EAR files with filenames that begin "wsgw" from the *WebSphere_DeployMgr_root/installableApps* directory of the machine on which you installed WebSphere Application Server Network Deployment into the *stand-alone_WAS_HOME/installableApps* directory of the target application server's install tree, where *stand-alone_WAS_HOME* is the root directory of your target application server (by default WebSphere/AppServer).
- Copy file *WebSphere_DeployMgr_root/lib/WSGWResourceBundles.jar* into directory *stand-alone_WAS_HOME/lib*.
- Copy directory *WebSphere_DeployMgr_root/WSGW* and all files and directories within it into directory *stand-alone_WAS_HOME/WSGW*.
- If you plan to use the Cloudscape database with the Web services gateway, then copy directory *WebSphere_DeployMgr_root/bin/WSGWDB* and all files and directories within it into directory *stand-alone_WAS_HOME/bin/WSGWDB*.

In this task you install the gateway into an individual application server instance in your network space. The major elements of this process are as follows:

- Database and Table creation (optional - DB2 only).
- JDBC driver and Datasource creation (optional - only if a database is being installed).
- Installation of the following enterprise applications:
 - The gateway application.
 - Apache SOAP channel 1.
 - The correlation application (optional - only if a database is being installed).
- Installation of other gateway applications. For example:
 - Apache SOAP channel 2.

To install the gateway into an application server instance, complete the following steps:

Steps for this task

1. **(Optional)** To create and install a DB2 database and associated tables, complete the following steps:
 - a. From a command prompt, go to directory *stand-alone_WAS_HOME/WSGW/scripts/install*.
 - b. Enter the command `WSGWinstallDB`:

- **[Windows]** WSGWinstallDB.bat <i>stand-alone_WAS_HOME</i>
<i>db2user_id</i> <i>db2password</i>
- **[UNIX]** WSGWinstallDB.sh <i>stand-alone_WAS_HOME</i>
<i>db2user_id</i>

For example (UNIX systems):

```
./WSGWinstallDB.sh /opt/WebSphere/AppServer mydb2id
```

Note: Running WSGWinstallDB also creates WSGWInstallDB.log in the stand-alone application server's /logs directory. Open this file to check that the database was created successfully.

2. Start the application server (you can use the command startServer<i>.ext</i> <i>your_server</i>).
3. From a command prompt, go to directory <i>stand-alone_WAS_HOME</i>/WSGW/scripts/install.
4. Clear your class path.

You can use the following command:

- (Windows systems): set CLASSPATH=
- (UNIX systems): unset CLASSPATH

5. Enter the command <i>stand-alone_WAS_HOME</i>/bin/wsadmin<i>.ext</i> -f setupWSGW.jac1 <i>parm1 ... parmN</i>

where

- *parm1* is the <i>stand-alone_WAS_HOME</i> directory
- *parm2* is the server name
- *parm3* is the node name (this is case sensitive)

If you are not using a database, there are no more parameters and wsgwcorr.ear is not installed. For example (Windows systems):

```
C:\Progra~1\WebSphere\AppServer\bin\wsadmin.bat -f setupWSGW.jac1
C:/Progra~1/WebSphere/AppServer server1 PHJ2
```

Note: The use of forward slashes (/) is compulsory for this command, even on Windows systems.

If you are using Cloudscape, there is one more parameter:

- *parm4* is the name and location of the WSGWDB directory (<i>stand-alone_WAS_HOME</i>/bin/WSGWDB).

For example (Windows systems):

```
C:\Progra~1\WebSphere\AppServer\bin\wsadmin.bat -f setupWSGW.jac1
C:/Progra~1/WebSphere/AppServer server1 PHJ2
C:/Progra~1/WebSphere/AppServer/bin/WSGWDB
```

Note: The use of forward slashes (/) is compulsory for this command, even on Windows systems.

If you are using DB2, there are four more parameters:

- *parm4* is WSGWDB.
- *parm5* is your DB2 user ID

- *parm6* is your DB2 password
- *parm7* is the name and location of file *db2java.zip*

For example (Windows systems):

```
C:\Progra~1\WebSphere\AppServer\bin\wsadmin.bat -f setupWSGW.jac1
C:/Progra~1/WebSphere/AppServer server1 PHJ2 WSGWDB mydb2id mydb2pw
C:/Progra~1/SQLLIB/java/db2java.zip
```

Note: The use of forward slashes (/) is compulsory for this command, even on Windows systems.

When you run the `setupWSGW.jac1` command, the following initial set of gateway applications is installed:

- The gateway itself (*wsgw.ear*).
 - Apache SOAP channel 1 (*wsgwsoap1.ear*).
 - (Optionally) The correlation application (*wsgwcorr.ear*).
6. Install additional gateway applications. For example:
- Apache SOAP channel 2 (*wsgwsoap2.ear*)

To install additional gateway applications, complete the following steps:

Note: If you prefer, you can install these EAR files using the WebSphere Application Server administrative console, as described in the final step of the task Enabling operation-level authorization.

- From a command prompt, go to directory `<i>stand-alone_WAS_HOME</i>/bin`
- To start the application server, enter the command `startServer<i>.ext</i> <i>your_server</i>`
- To start the WebSphere administration program, enter the command `wsadmin<i>.ext</i>`.
- For each additional Web services gateway enterprise application that you want to install, enter the following commands at the `wsadmin>` prompt:


```
$AdminApp install path_to_ear_file {-appName application -server your_server
                                     -node your_node_name}
$AdminConfig save
where
```

 - *application* is the name of the enterprise application
 - *path_to_ear_file* is the name and location of the enterprise application's EAR file
 - *your_node_name* is the node name (this is case sensitive)

For example (Windows systems):

```
$AdminApp install C:/Progra~1/WebSphere/AppServer/installableApps
/wsgwsoap2.ear {-appName wsgwsoap2 -server server1 -node PHJ2}
$AdminConfig save
```

- After you have installed all your additional Web services gateway enterprise applications, close the WebSphere administration program by entering `quit` or `exit` at the `wsadmin>` prompt.
7. To stop then restart the application server, complete the following steps:
- Enter the command `stopServer<i>.ext</i> <i>your_server</i>`
 - Enter the command `startServer<i>.ext</i> <i>your_server</i>`

What to do next

You are now ready to test the installation.

Testing the Web services gateway installation

Use this task to test that the Web services gateway has been installed correctly.

To test the basic installation of the Web services gateway, complete the following steps:

Steps for this task

1. In a Web browser, go to `http://<i>host</i>:<i>port</i>/wsgw` where *host* and *port* are the host name and port number that your HTTP server is listening on.

The browser should display the following message:

IBM Web services gateway

What do you want to do?

- Run the admin client
- View the product ID

Backing up and restoring a gateway configuration

The **Back up** options write out, to a single file, the deployment details for all the channels, filters, UDDI references and Web services that are currently deployed to the gateway. The **Restore** option uses the information contained in a previously-saved file to populate an empty installation of the gateway with the same deployment details for those channels, filters, UDDI references and Web services.

There are two situations in which you might want to back up a gateway configuration:

- Before you apply an upgrade or fix pack to WebSphere Application Server.
- Because you want to create or update a gateway cluster.

When you *apply an upgrade or fix pack*, the configured gateway is replaced with an upgraded but empty gateway. You use the **Save Gateway Configuration -> Private** option to preserve your configuration.

When you *want to create or update a gateway cluster*, you use the **Save Gateway Configuration -> Shared** option to save a "shareable" version of your gateway configuration. This saved configuration does not include machine-specific values, and therefore does not over-write these values on the target gateways in your cluster when you restore the configuration to them.

The **restore** option automatically detects whether the configuration file that it is restoring contains a **Private** or a **Shared** gateway configuration.

Note: For the **Save Gateway Configuration -> Private** option, the steps below include

- Manual un-publishing of Web services from UDDI before backing up the gateway configuration.
- Manual re-publishing of these Web services to UDDI after restoring the gateway configuration.

The reasons for this are explained in Backing up and restoring UDDI publication links.

To back up and restore a gateway configuration, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. **(Optional)** If you are backing up before you apply an upgrade or fix pack, and any of your deployed Web services are also published by the gateway to UDDI registries, use the Listing and managing gateway-deployed Web services option to un-publish them from UDDI. Make a note of the UDDI deployment details so that you can re-publish them to UDDI after you restore the gateway configuration.
3. To **back up** the gateway configuration, complete the following steps:

a.

- 1) In the gateway user interface navigation pane, click the following link:

Gateway

- Back Up

The **Save Gateway Configuration** screen is displayed.

- 2) In the **Location** field, type the path to your configuration file.

Note:

- The path to your configuration file must point to a local drive on the machine on which the gateway is currently running.
- The file name for the configuration file can be any valid Java filename.

- 3) If you are backing up before you apply an upgrade or fix pack, select **Private**. If you are backing up to create a gateway cluster, select **Shared**.
- 4) Click **OK**

Your gateway configuration is saved to the location you specified.

- b. To **restore** a gateway configuration, complete the following steps:

- 1) Check that the target gateway is empty. If there are any channels, filters, UDDI references or Web services deployed to the gateway, then remove them.

Note: If you have just installed a WebSphere Application Server fix pack or upgrade program, then the upgraded gateway is empty.

- 2) In the gateway user interface navigation pane, click the following link:

Gateway

- Restore

The **Restore Gateway Configuration** screen is displayed.

- 3) In the **Location** field, type the path to your configuration file.

Note:

- The path to your configuration file must point to a local drive on the machine on which the gateway is currently running.
- The saved file is not specific to a given version of the gateway.
- If you are restoring a file that was backed up using the **Save Gateway Configuration** -> **Private** option, then the backup contains deployment details (for example URIs) that are specific to a given

gateway instance. If you restore this type of backup file to a different instance (for example a gateway on a host with a different network identity), then the restore will succeed but you will have to use the gateway user interface to correct any resultant errors in the deployment details.

4) Click **OK**

Your gateway configuration is restored to the gateway that you are currently administering.

- c. **(Optional)** If you have just restored a configuration after applying an upgrade or fix pack, and any of the deployed Web services were previously published to UDDI registries, use the Listing and managing gateway-deployed Web services option (and the notes you made in step 2 above) to re-publish these Web services to UDDI.

Backing up and restoring UDDI publication links

When you save a gateway configuration, the deployment details through which a deployed Web service has been published to UDDI by the gateway are not saved. The reasons for not backing up these details are as follows:

- When a gateway is replaced through a fix pack or upgrade program, or otherwise removed without first using the gateway administrative console to un-publish each Web service from UDDI, then the associated entries in UDDI registries are not removed.
- When a gateway is restored, if the UDDI publication details were also restored, then the Web services would be re-published to UDDI by the gateway.
- When a gateway publishes the same service twice to a UDDI registry, the registry does not overwrite the initial publication. It creates a second copy of the TModel and Service definition.
- When a gateway removes a service from a UDDI registry, it only removes the last one published. So it can leave behind defunct TModels and Service Definitions.

So, to preserve the integrity of the UDDI registries, restoring the gateway does not automatically re-publish any Web services to UDDI. But if you know that a service has been un-published from UDDI (for example because you removed it manually before backing up the gateway configuration) then you can safely re-publish it to UDDI after the configuration has been restored.

Creating and updating a gateway cluster

In order to improve performance and availability, you configure WebSphere Application Server so that a set of application servers acts as a cluster. In a cluster, several application servers are configured identically, and a plug-in to the front-end HTTP server shares incoming HTTP requests between the available application servers in the cluster. Application servers in a cluster can each be hosted on a separate machine, or all hosted on one or two machines. For a more detailed overview of options for clustering in WebSphere Application Server Network Deployment, see Setting up a multinode environment.

To create a *gateway* cluster, you manually install an identically-configured gateway on every application server in an existing cluster.

To create and update a gateway cluster, complete the following steps:

Steps for this task

1. Create a cluster of application servers. For example, by completing the following substeps:

Note: These substeps illustrate how to use the WebSphere Application Server Network Deployment deployment manager and administrative console to create a two-server cluster within a cell. In this example, each clustered application server is on a separate node of the cell. However there are other valid configurations, and other ways of creating a cluster. For more detailed information see the InfoCenter for WebSphere Application Server Edge components (<http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html>), which contains complete documentation for the Caching Proxy and the Load Balancer in the following PDF online books: *WebSphere Application Server Concepts, Planning, and Installation for Edge Components*; the *WebSphere Application Server Caching Proxy Administration Guide*, and the *WebSphere Application Server Programming Guide for Edge Components*.

- a. Install WebSphere Application Server on machines X and Y.
- b. Install WebSphere Application Server Network Deployment on machine X.
- c. Stop all application servers on both machines.
- d. Add machines X and Y as nodes in a deployment manager cell by entering the following command from a command line *on each machine*:

```
WAS_HOME/bin/addnode dm_machine_hostname dm_port -includeapps  
where
```

- *WAS_HOME* is the root directory of your target application server (by default WebSphere/AppServer).
- *dm_machine_hostname* is the host name for the machine on which the deployment manager for this cell is running (in this case machine X).
- *dm_port* is the port on which the deployment manager is listening (by default 8879).

For example (Windows systems):

```
C:\Progra~1\WebSphere\AppServer\bin\addnode xhost 8879 -includeapps
```

- e. Use the WebSphere Application Server Network Deployment administrative console to create new, clustered application servers on the two machines:
 - 1) Select **Servers -> Clusters -> New**.
 - 2) Enter a name for the cluster.
 - 3) Clear the **Prefer local enabled** check box.
 - 4) Select the **Do not include an existing server in this cluster** check box.
 - 5) Create a new clustered application server on machine X (enter a name for the application server; select the node for this application server (machine X); clear the **generate unique HTTP Ports** check box; click **Apply**).
 - 6) Repeat the previous step to create a new clustered application server on machine Y.
 - 7) Click **Next**.
 - 8) Click **Finish**.

The new cluster is created.

2. Install a gateway on every application server in the cluster.
3. Select and configure the gateway that is to be the source configuration for every gateway in the cluster.

Note:

- The gateway that is chosen as the source for the cluster configuration need not be part of the cluster.
 - The channels deployed to this gateway must have their End Point addresses configured for clustering. See Deploying channels to the Web services gateway.
4. Use the back up and restore options to save the source configuration, then restore it to every gateway in the cluster.
 5. Apply all subsequent gateway updates only to the source configuration gateway, then use the back up and restore options to restore the updated configuration to every gateway in the cluster.

Note: This is particularly important with regard to Web services that have been published to UDDI by the gateway. When you save a gateway configuration, the record of which Web services have been gateway-published to UDDI is not saved (for reason that are explained in Backing up and restoring UDDI publication links). So if you change the gateway that you use as the source for cluster updates, you lose the record of which Web services have been gateway-published to UDDI.

Administering the Web services gateway

Before you begin

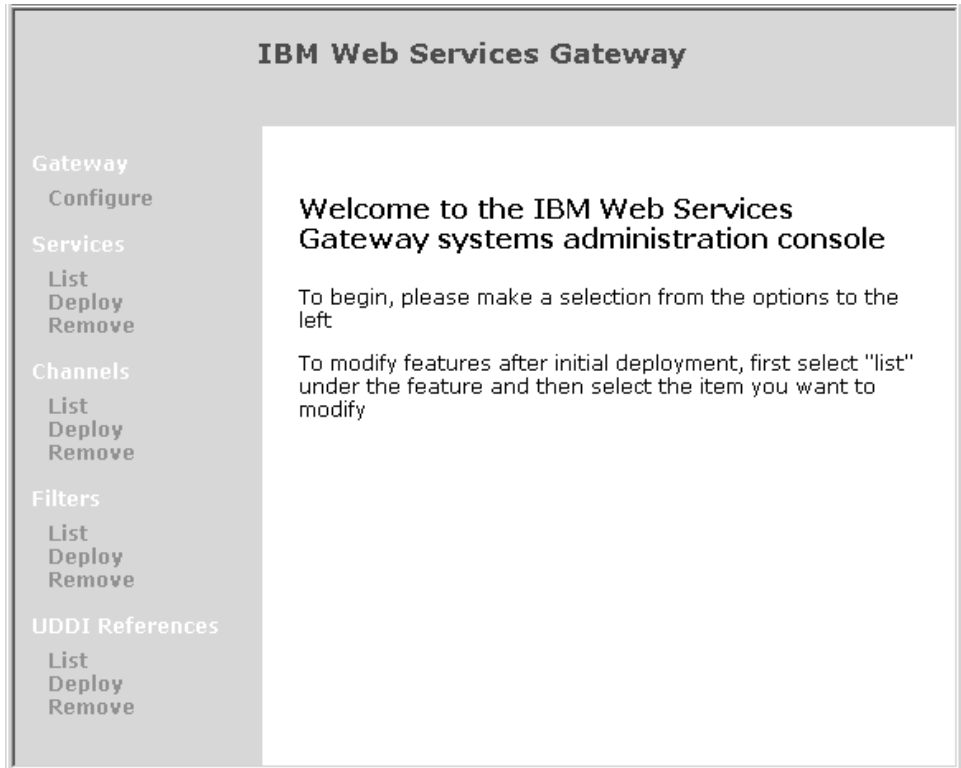
To administer the Web services gateway, complete the following steps:

Steps for this task

1. Start the WebSphere Application Server Administrative Server.
2. Open the following Web page:
`http://<i>host</i>:<i>port</i>/wsgw/admin/index.html`

where *host* and *port* are the host name and port number that your HTTP server is listening on. For example `localhost:8080` or `localhost:9080`.

The main administration page is displayed:



The order of the elements on this page is significant, for the following reasons:

- If you change the namespace URI or WSDL URI (using the **Configure Gateway** option), you break the link back to the gateway for every Web service that you have already deployed. So you must set these URIs before you deploy any Web services.
- When you configure a Web service, you choose the following entities:
 - The channels on which it is available.
 - The filters (if any) which apply to it.
 - The UDDI references (if any) to which it is deployed.

Each of these choices is made from a list of resources which have already been deployed to the Web services gateway. So you might want to deploy your channels, filters and UDDI references to the gateway before you configure the Web services that use them.

3. For more information on how to configure each element of the Web services gateway, see the following topics:
 - Setting the namespace URI and WSDL URI for the Web services gateway
 - Working with channels
 - Working with filters
 - Working with UDDI references
 - Working with Web services

Note:

- You configure each of the above elements of the gateway by filling in fields in a panel.
- In all of the gateway panels, fields marks with asterisks are required.

- After you deploy a channel, filter, or UDDI reference you should refresh all other open browser windows to ensure that up-to-date lists are displayed.

Setting the namespace URI and WSDL URI for the Web services gateway

Before you begin

Initial values for the namespace URI and WSDL URI are automatically configured when you install the Web services gateway.

When you deploy a Web service to the Web services gateway, these two URIs are used as follows:

- The **Namespace URI for services** is used as the namespace for the gateway services in exported WSDL documents.
- The **WSDL URI for exported definitions** is used to generate the URL in import statements within exported WSDL documents.

Note: When you change these URIs, you break the link back to the Web services gateway for every Web service that you have already deployed. So you must set these URIs before you deploy any Web services to the Web services gateway.

To set the namespace URI and WSDL URI for the Web services gateway, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Gateway

- Configure

The gateway configuration form is displayed:

The screenshot shows the IBM Web Services Gateway administrative interface. The main window is titled "Configure Gateway". On the left, there is a navigation pane with the following items: Gateway (selected), Configure (highlighted), Services, List, Deploy, Remove, Channels, List, and Denlov. The main content area contains two text input fields: "Namespace URI for services" with the value "urn:ibmwsgw" and "WSDL URI for exported definitions" with the value "http://PHJ2:9080/wsgw". Below these fields is an "Apply Changes" button.

3. In the **Namespace URI for services** field, type the new name.

There is no fixed syntax for the namespace URI, but whatever name you choose is likely to be more effective if it observes the following guidelines:

- It begins with "urn:"

See the guidance on Internet standards for the syntax of Uniform Resource Names (URNs) at

<http://www.ietf.org/rfc/rfc2141.txt>(<http://www.ietf.org/rfc/rfc2141.txt>).

- It is globally unique.
- It reflects your company name.

4. In the **WSDL URI for exported definitions** field, type the new name.

The initial value is the gateway's "best guess" at the right value, but you will probably want to overwrite it with a new value. For example it might guess a local URI such as `http://h1dswrth:9080/wsgw`, and because you are giving the WSDL to people in other companies you modify this to

`http://h1dswrth.your_company.com/wsgw`. Note that only the *host* and *port* parts of the initial value are modified, and that this URI should always start `http://` and end `/wsgw`.

5. Click **Apply Changes**.

What to do next

Working with channels

Before you begin

Before you can work with a channel, you must install the channel application in WebSphere Application Server as described in the penultimate step of Installing the gateway into a deployment manager cell and Installing the gateway into a stand-alone application server.

Two versions of each type of channel are supplied so that, for each channel type, you can set up separate channels for inbound and outbound requests. For more information see Channels - entry points to the Web services gateway.

From the navigation pane of the Web services gateway administrative user interface, you can choose the following actions for **Channels**:

- **List** to list the deployed channels, and modify their deployment details.
- **Deploy** to deploy a channel.
- **Remove** to remove channels.

Channels - entry points to the Web services gateway

Channels form entry points to the Web services gateway and carry requests and responses between Web services and the Web services gateway. A request to the Web services gateway arrives through a channel, is translated into a WSIF message, then passed through any filters that are registered for the requested service, and finally sent on to the service implementation. Responses follow the same path in reverse.

Before you can use a channel, you must install the channel application in WebSphere Application Server then deploy the channel to the Web services gateway.

Note: A deployed channel is not used until you deploy a Web service that uses the channel.

Two versions of each type of channel are supplied with the gateway. This is so that, for each channel type, you can set up separate channels for inbound and outbound requests. This provides a simple mechanism for giving different access rights to users from outside your organization from the rights you give to users within your organization:

- To ensure that users outside your organization can only access those internal services that you choose to publish externally, you deploy those services on the inbound channel.
- To give users inside your organization access to the full range of internal and external services, you deploy those services on the outbound channel.

Listing and managing gateway-deployed channels

Use this task to list the channels that are deployed to the Web services gateway, and modify their deployment details.

To list the channels that are currently deployed to the Web services gateway, and view and modify their deployment details, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Channels

- List

The main pane is updated with a list of all the channels that are deployed to the Web services gateway.

3. Click the name of a channel in the list. A form is displayed through which you can view and modify the current deployment details for this channel.
4. Modify the following deployment details:

Home Location

Type the name of the new home for this channel.

End Point Address

Type the new address on which the channel is to listen.

5. **(Optional)** If this channel is intended to be used to receive asynchronous reply messages, type appropriate values in the following two fields. Otherwise leave them blank.

Note: If the channel supports asynchronous messaging, then the deployment details documentation for the channel should indicate what values to enter in these fields.

Async Reply Context Name

Async Reply Context Value

6. To start this channel, enable the **YES** radio button. To stop this channel, enable the **NO** radio button.
7. Click **Apply changes**.

Results

If the processing completes successfully, the list of deployed channels is redisplayed. Otherwise, an error message is displayed.

Deploying channels to the Web services gateway

Before you begin

Before you can deploy a channel, you must install the channel application in WebSphere Application Server as described in the penultimate step of Installing the gateway into a deployment manager cell and Installing the gateway into a stand-alone application server.

If you want to deploy the channels supplied with the Web services gateway, their deployment details are listed in Web services gateway - Channel deployment details.

To deploy a channel, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Channels

- Deploy

A form is displayed for you to specify the deployment details.

3. Type the following channel deployment details:

Channel Name

Type the name by which the channel will be known within the Web services gateway (and by which it will be listed using the **Channels > List** option). This name must be unique within the gateway.

Home Location

Type the name of the home for this channel.

End Point Address

Type the address on which the channel is to listen.

4. **(Optional)** If this channel is intended to be used to receive asynchronous reply messages, type appropriate values in the following two fields. Otherwise leave them blank.

Note: If the channel you are deploying supports asynchronous messaging, then the deployment details documentation for the channel should indicate what values to enter in these fields.

Async Reply Context Name

Async Reply Context Value

5. Click **OK**.

Results

If the processing completes successfully, the list of deployed channels is updated to include the new channel. Otherwise, an error message is displayed.

What to do next

Note: The deployed channel will not be used until you deploy a Web service that uses the channel.

Web services gateway - Channel deployment details

The deployment details for the channels supplied with the Web services gateway are listed below:

- Apache SOAP channel 1
 - **Channel Name:** ApacheSOAPChannel1
 - **Home Location:** ApacheSOAPChannel1Bean
 - **Async Reply Context Name:** Leave blank. This function is not supported by this channel.
 - **Async Reply Context Value:** Leave blank. This function is not supported by this channel.
- Apache SOAP channel 2
 - **Channel Name:** ApacheSOAPChannel2
 - **Home Location:** ApacheSOAPChannel2Bean
 - **Async Reply Context Name:** Leave blank. This function is not supported by this channel.
 - **Async Reply Context Value:** Leave blank. This function is not supported by this channel.

Removing channels from the Web services gateway

To remove a channel, complete the following steps:

Steps for this task

1. Display the Web Services gateway administrative user interface.
2. In the navigation pane, click the following link:

Channels

- Remove

The main pane is updated with a list of all the channels that are deployed to the Web services gateway. Alongside each entry in the list is a check box, and information on the number of Web services that currently use the channel.

3. **(Optional)** Click the name of a channel in the list.

A form is displayed through which you can view the current deployment details for this channel, including a list of the Web services that currently use the channel.

4. Select the check box for every channel that you want to remove.

Note: When you remove a channel that is currently used by one or more Web services, the gateway removes the channel from the channel list for each associated Web service.

5. Click **OK**.

Results

If the processing completes successfully, the list of deployed channels is updated. Otherwise, an error message is displayed.

Working with filters

Before you begin

Before you can work with a filter, you must install the filter application in WebSphere Application Server as described in the penultimate step of Installing the gateway into a deployment manager cell and Installing the gateway into a stand-alone application server.

From the navigation pane of the Web services gateway administrative user interface, you can choose the following actions for **Filters**:

- **List** to list the deployed filters, and modify their deployment details.
- **Deploy** to deploy a filter.
- **Remove** to remove filters.

Filters - service interceptors for the Web services gateway

Filters are used to intercept service invocations which come into the Web services gateway, and responses which leave it. Filters can perform a wide range of tasks, from logging messages, to transforming their content, to terminating an incoming request. Filters are deployed to the Web services gateway as described in Deploying filters to the Web services gateway, then registered for use with individual Web services as described in "Working with Web services".

Listing and managing gateway-deployed filters

Use this task to list the filters that are deployed to the Web services gateway, and modify their deployment details.

To list the filters that are currently deployed to the Web services gateway, and view and modify their deployment details, complete the following steps:

Steps for this task

1. Display the Web Services gateway administrative user interface.
2. In the navigation pane, click the following link:

Filters

- List

The main pane is updated with a list of all the filters that are deployed to the Web services gateway.

3. Click the name of a filter in the list. A form is displayed through which you can view and modify the current deployment details for this filter.
4. Modify the following deployment detail:

Home Location

Type the name of the new home for this filter.

5. Click **Apply changes**.

Results

If the processing completes successfully, the list of deployed filters is redisplayed. Otherwise, an error message is displayed.

Deploying filters to the Web services gateway

Before you begin

Use this task to deploy a filter to the Web services gateway.

Note: The deployed filter will not be used until you deploy a Web service that uses the filter.

Before you can deploy a filter, you must install the filter application in WebSphere Application Server as described in the penultimate step of Installing the gateway into a deployment manager cell and Installing the gateway into a stand-alone application server.

Note: You can deploy multiple instances of a filter by entering different filter names.

To deploy a filter, complete the following steps:

Steps for this task

1. Display the Web Services gateway administrative user interface.
2. In the navigation pane, click the following link:

Filters

- Deploy

A form is displayed for you to specify the deployment details.

3. Type the following filter deployment details:

Filter Name

Type the name by which the filter will be known within the Web services gateway (and by which it will be listed using the **Filters > List** option). This name must be unique within the Web services gateway.

Home Location

Type the name of the home for this filter.

4. Click **OK**.

Results

If the processing completes successfully, the list of deployed filters is updated to include the new filter. Otherwise, an error message is displayed.

Removing filters from the Web services gateway

To remove a filter, complete the following steps:

Steps for this task

1. Display the Web Services gateway administrative user interface.
2. In the navigation pane, click the following link:

Filters

- Remove

The main pane is updated with a list of all the filters that are deployed to the Web services gateway. Alongside each entry in the list is a check box, and information on the number of Web services that currently use the filter.

3. **(Optional)** Click the name of a filter in the list.

A form is displayed through which you can view the current deployment details for this filter, including a list of the Web services that currently use the filter.

4. Select the check box for every filter that you want to remove.

Note: When you remove a filter that is currently used by one or more Web services, the gateway removes the filter from the filter lists for each associated Web service.

5. Click **OK**.

Results

If the processing completes successfully, the list of deployed filters is updated. Otherwise, an error message is displayed.

Working with UDDI references

A UDDI reference is a pointer to a UDDI registry. This may be a private UDDI registry such as the (IBM WebSphere UDDI Registry), or a public UDDI registry.

In the UDDI model, Web services are owned by "businesses", and "businesses" are owned by "Authorized Names". Each UDDI reference gives access to the Web services that are owned by a single "Authorized Name" in single UDDI registry.

From the navigation pane of the Web services gateway administrative user interface, you can choose the following actions for **UDDI References**:

- **List** to list the deployed UDDI references, and modify their deployment details.
- **Deploy** to deploy a UDDI reference.
- **Remove** to remove UDDI references.

For more information about how the gateway works with UDDI registries, see [UDDI registries - Web service directories that integrate with the Web services gateway and Publishing a Web service to a UDDI registry for deployment to the gateway](#). For more general information about UDDI and UDDI registries, see the UDDI community at uddi.org(<http://uddi.org>).

UDDI registries - Web service directories that integrate with the Web services gateway

UDDI

The Universal Description, Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web services.

In this specification:

- Each Web service is owned by one "business", and each "business" (and the Web services it owns) is maintained by one "Authorized Name".
- One "Authorized Name" can own many "businesses", and one "business" can own many Web services.

The UDDI specification also associates Web services with "Technical models". These are generic categories that allow a UDDI registry user to search for a *type* of service, rather than needing to know the access details for a specific service.

For more general information about UDDI, see the UDDI community at uddi.org(<http://uddi.org>).

UDDI registries

UDDI registries use the UDDI specification to publish directory listings of Web services. There are Universal Business Registries (sometimes referred to as 'public UDDI registries') hosted worldwide, including one hosted by IBM. Enterprises can also host their own internal registries behind their firewalls (sometimes referred to as 'private UDDI registries') to better manage their internal implementation of Web services. The (WebSphere UDDI Registry) is an example of a private UDDI registry.

How the gateway interacts with UDDI registries

The gateway interacts with UDDI registries in two ways:

- When you deploy a Web service to the gateway, you specify the location of the "internal" WSDL file that describes the Web service to be deployed. This WSDL file can be located through a UDDI registry.
- For any gateway-deployed Web service, you can tell the gateway to create entries for the Web service in one or more UDDI registries.

To enable your gateway to interact with a UDDI registry, you create one or more gateway pointers to the registry. The gateway refers to these pointers as "UDDI references", and you create them as described in Deploying UDDI references to the Web services gateway. Each UDDI reference includes the following parameters:

- The access points for the UDDI registry (the **Inquiry URL** and the **Publish URL**).
- The "Authorized Name" (the **User ID** and **Password**) for the owner of one or more "businesses" in the UDDI registry.

You get the "Authorized Name" from the target UDDI registry. For more information see Publishing a Web service to a UDDI registry for deployment to the gateway.

Note: A given UDDI reference can only access the Web services that are owned by the "businesses" that are in turn owned by a single "Authorized Name". So if you need access to two Web services in the same registry, and each service is owned by a different "Authorized Name", then you need to create two UDDI references.

When you deploy a Web service, and you specify that the "internal" WSDL file is located through a UDDI registry, you enter the following two parameters:

- The UDDI reference that can access this service.
- The "service key" that the UDDI registry has assigned to this service.

You get the "service key" from the target UDDI registry. For more information see Publishing a Web service to a UDDI registry for deployment to the gateway

When you tell the gateway to create entries for a deployed Web service in one or more UDDI registries, you enter the following two parameters:

- The UDDI references (one for each registry) that can access the UDDI business category under which you want to publish this service.
- The "business key" that identifies the UDDI business category.

You get the "business key" from the target UDDI registry. For more information see [Publishing a Web service to a UDDI registry for deployment to the gateway](#)

Note: Because the gateway only interacts with UDDI registries at the level of specific Web services, the gateway does not make use of UDDI "technical models".

Listing and managing gateway-deployed UDDI references

Use this task to list the UDDI references that are deployed to the Web services gateway, and modify their deployment details.

To list the UDDI references that are currently deployed to the Web services gateway, and view and modify their deployment details, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

UDDI References

- [List](#)

The main pane is updated with a list of all the UDDI references that are deployed to the Web services gateway.

3. Click the name of a UDDI reference in the list. A form is displayed through which you can view and modify the current deployment details for this UDDI reference.
4. Modify the following deployment details:

Inquiry URL

Type the new URL that provides access to this registry for the SOAP inquiry API.

Publish URL

Type the new URL that provides access to this registry for the SOAP publish API.

User Name

Type the new user ID for the "Authorized Name" that has update access to this registry.

Password

Type the password for this new user ID.

Confirm Password

Type again the password for this new user ID.

Note:

The values you enter here for **User Name** and **Password** must match those of the owner of the corresponding business in UDDI. You can see the owning user ID in UDDI by looking at the business details under the "Authorized Name" field.

If the values you enter here do not match the "Authorized Name" values for the business that owns the service, then the service will not be published or found.

If the business that owns the service has more than one "Authorized Name", you might want to set up multiple UDDI references (each with a different user ID) to the same UDDI registry .

5. Click **Apply changes**.

Results

If the processing completes successfully, the list of deployed UDDI references is redisplayed. Otherwise, an error message is displayed.

Deploying UDDI references to the Web services gateway

Before you begin

Note: The deployed UDDI reference will not be used until you deploy a Web service that uses the UDDI reference.

To deploy a UDDI reference, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

UDDI References

- [Deploy](#)

A form is displayed for you to specify the deployment details.

3. Type the following UDDI reference deployment details:

Reference Name

Type the name by which the UDDI reference will be known within the Web services gateway (and by which it will be listed using the **UDDI References > List** option). This name must be unique within the gateway - and note that you might need more than one UDDI reference for a given UDDI registry (for more information see UDDI registries - Web service directories that integrate with the Web services gateway).

Inquiry URL

Type the URL that provides access to this registry for the SOAP inquiry API.

Publish URL

Type the URL that provides access to this registry for the SOAP publish API.

User Name

Type the user ID for an "Authorized Name" that has update access to this registry.

Password

Type the password for this user ID.

Confirm Password

Type again the password for this user ID.

Note:

The values you enter here for **User Name** and **Password** must match those of the "Authorized Name" in the UDDI registry. You can see the owning user ID in UDDI by looking at the business details under the "Authorized Name" field.

4. Click **OK**.

Results

If the processing completes successfully, the list of deployed UDDI references is updated to include the new UDDI reference. Otherwise, an error message is displayed.

Removing UDDI references from the Web services gateway

To remove a UDDI reference, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

UDDI References

- Remove

The main pane is updated with a list of all the UDDI references that are deployed to the Web services gateway. Alongside each entry in the list is a check box, and information on the number of Web services that currently use the UDDI reference.

3. **(Optional)** Click the name of a UDDI reference in the list.

A form is displayed through which you can view the current deployment details for this UDDI reference, including a list of the Web services that currently use the UDDI reference.

4. Select the check box for every UDDI reference that you want to remove.

Note: When you remove a UDDI reference that is currently used by one or more Web services, the gateway removes the UDDI reference from the UDDI reference list for each associated Web service.

5. Click **OK**.

Results

If the processing completes successfully, the list of deployed UDDI references is updated. Otherwise, an error message is displayed.

Working with Web services

Before you begin

If you change the Namespace URI, you break the link back to the Web services gateway for every Web service that you have already deployed. So you must set the Namespace URI before you deploy any Web services.

When you configure a Web service, you choose the following resources:

- The channels on which the service is available.
- The filters (if any) that apply to the service.

- The UDDI registries (if any) in which entries for the service are created.

Each of these choices is made from a list of resources that have already been deployed to the Web services gateway. So you must deploy your channels, filters and references to UDDI registries to the Web services gateway before you deploy the Web services that use those resources.

From the navigation pane of the Web services gateway administrative user interface, you can choose the following actions for **Services**:

- **List** to list the deployed Web services, and modify their deployment details.
- **Deploy** to deploy a Web service.
- **Remove** to remove Web services.

Listing and managing gateway-deployed Web services

Use this task to list the Web services that are deployed to the Web services gateway, and modify their deployment details.

Before you begin

There is no point in deploying multiple target services to the same gateway service unless you have a filter implementation that is capable of selecting the required target service.

To list the Web services that are currently deployed to the Web services gateway, and view and modify their deployment details (including adding or removing multiple target services) complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Services

- List

The main pane is updated with a list of all the Web services that are deployed to the Web services gateway.

3. Click the name of a Web service in the list. A form is displayed through which you can view and modify the current deployment details for this Web service, and add or remove multiple target services.
4. At the level of the gateway service itself (in the **Gateway Service Properties** section) you can change the following settings. When you have finished making changes, click **Apply Changes**.
 - a. **Authorization Policy - Control access to this service.** Use this check box to enable or disable operation-level authorization for this gateway service.
 - b. **Audit Policy - Log requests to this service.**

The Audit policy indicates whether the MessageWarehouse object, if present, should be used to log requests and responses for this service. If you have a Message Warehouse implementation, use this check box to enable or disable logging of requests and responses for this Web service.
 - c. In this release of the gateway, the **Annotation URL** field is not used.
 - d. If you want to publish the service to one or more UDDI registries (selected in the **UDDI References** section below), enter the UDDI business key in

the field provided under **UDDI Publication Properties**. This key identifies the business category under which you want your service to appear in UDDI. To get a list of valid business keys, look up businesses in a UDDI registry. This is an example of a UDDI business key: 08A536DC-3482-4E18-BFEC-2E2A23630526. For more information about UDDI business keys see Publishing a Web service to a UDDI registry for deployment to the gateway.

5. In the **Target Services** section you can add, modify or remove services from a list of target services for this single gateway service. Every service on this list provides exactly the same service, and they are presented by the gateway to the service requesters as a single gateway service. To add a new target service, complete the following steps:

- a. **WSDL Location.** Type the location of the "internal" WSDL file that describes the Web service to be deployed. The WSDL file is either located at a URL, or through a UDDI registry.

If the WSDL location is a URL, type the URL (if the binding and service definition for this Web service are held in separate WSDL files, then type the URL of the WSDL file that defines the binding).

If the WSDL is located through a UDDI registry, type `<i>uddiReference</i>`, `<i>serviceKey</i>` where

- *uddiReference* is the reference name by which a currently-deployed UDDI reference is known within the gateway (and by which it is listed using the **UDDI References -> List** option)
- *serviceKey* is the service key that the UDDI registry has assigned to the service. This is an example of a UDDI service key: 34280367-0ECF-46CE-B804-14C21D6D0FB1. For more information about UDDI service keys see Publishing a Web service to a UDDI registry for deployment to the gateway.

Note:

- When the Web services gateway deploys the Web service, it generates a matching "external" WSDL file that it makes available to gateway users. This "external" WSDL file also describes the service, but is located at a new URL and is generated and maintained by the Web services gateway itself.
- If the Web service is also published to one or more UDDI registries, then the "internal" WSDL file is required to remove the service from the gateway.

- b. **Location Type.** Select the type of location (either URL or UDDI) where the "internal" WSDL file is held.
- c. **Target Service Name.** If the Web service WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the target service's name from the target service WSDL.
- d. **Target Service Namespace.** If the Web service WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the namespace of the target service's name from the target service WSDL.
- e. **Target Service Identity Information.** Type the identity by which the target service is known within the Web services gateway. This identity need not be unique.
- f. Click **add**.

The target service is added to a list of target services.

6. In the **Channels** section, you can add or remove channels from the list of deployed channels through which this service is available.
7. In the **Request Filters** section, you can add or remove filters from the list of deployed filters that are applied to the request.
Note: The filters are executed in the order shown. To add a filter into the list at a particular position, use the **at position** menu.
8. In the **Response Filters** section, you can add or remove filters from the list of deployed filters that are applied to the response.
Note: The filters are executed in the order shown. To add a filter into the list at a particular position, use the **at position** menu.
9. In the **UDDI References** section, you can add or remove UDDI references from the list of deployed UDDI references to UDDI registries in which this service is published. If you select one or more UDDI references in this step, you must also enter a UDDI business key in the field provided under **UDDI Publication Properties** as described above. For more information about how the gateway works with UDDI registries, see UDDI registries - Web service directories that integrate with the Web services gateway.
10. In the **Exported WSDL definitions** section there are two pairs of WSDL links. Both pairs link to (a) the external WSDL implementation definition, and (b) the external WSDL interface definition.
 - To view details of the associated external WSDL for the service, use the first pair.
 - To return the WSDL for use by application programs that need the WSDL definitions for the service, use the second pair.

If there is an error generating the WSDL then a blank page is returned.

Note: To help your service users locate the WSDL documents for services that are deployed to the Web services gateway, the gateway also supports the WS-Inspection specification(<http://www.ibm.com/developerworks/webservices/library/ws-wsilspec.html>). To open a WS-Inspection document which contains references to the WSDL documents for all of the gateway-deployed services, you issue an HTTP GET against

```
http://host:port/wsgw/wsinspection.wsil
```

where *host* and *port* are the host name and port number that your HTTP server is listening on.

Deploying Web services to the Web services gateway

Before you begin

To deploy a Web service, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Services

- Deploy

A form is displayed for you to specify the deployment details.

3. Type the following Web service deployment details:
 - a. **Gateway Service Name.** Type the name by which the Web service will be known within the gateway, and by which it will be listed using the Services > List option. This name must be unique within the gateway and must not contain any spaces.

Note: If you have several implementations of the same Web service, you can map them all to the same deployed gateway service. You use the **Deploy** option (this option) to deploy just one instance of a given gateway service. To add more target services to a deployed gateway service, you use the Services > List option.
 - b. Choose between the two types of **Message part representation**. For more information see Data type representation - choosing between Generic classes and Deployed Java classes.

Generic classes

You can select this option in all cases except the following cases:

- You cannot select this option if your service has EJB or Java bindings that are used at runtime.
- You cannot select this option if your service uses Vectors, Enumerations, Hashtables or Maps.

Deployed Java classes

You must select this option if the target services for the gateway service use Vectors, Enumerations, Hashtables or Maps, or contain EJB or Java bindings.

Note: For Web services deployed with Java bindings (or EJB bindings even where the Web service is on a different server) you must also make sure that the specific Java classes that have been generated for the Web service are available to the gateway. For more information see Deploying Web services with Java bindings.

You can also select this option (as an alternative to selecting "Generic classes"):

- If the target service uses only supported simple and compound types.
- If the target service uses complex types, and the specific Java classes that have been generated for the Web service are available to the gateway (but note that if you select "Generic classes", then you don't need to also deploy the associated Java classes locally).

- c. **(Optional) Authorization Policy - Control access to this service.** If you want to enable operation-level authorization for this Web service, enable this check box.
- d. **(Optional) Audit Policy - Log requests to this service.**

The Audit policy indicates whether the MessageWarehouse object, if present, should be used to log requests and responses for this service. If you have a Message Warehouse implementation, and you want it to log requests and responses for this Web service, enable this check box.
- e. In this release of the gateway, the **Annotation URL** field is not used.
- f. Select the deployed resources that the Web service is to use, from the following lists:

Channels

Select one or more deployed channels through which this service is to be available.

Request Filters

Select zero or more deployed filters to apply to the request.

Response Filters

Select zero or more deployed filters to apply to the response.

UDDI References

Select zero or more deployed UDDI references (one for each UDDI registry) that can access the UDDI business category under which you want to publish this service. If you select one or more UDDI references in this step, you must also enter the UDDI business key in step 3h below. For more information about how the gateway works with UDDI registries, see *UDDI registries - Web service directories that integrate with the Web services gateway*.

- g. Specify the **Target Service Properties** for the Web service:

WSDL Location

Type the location of the "internal" WSDL file that describes the Web service to be deployed. The WSDL file is either located at a URL, or through a UDDI registry.

If the WSDL location is a URL, type the URL (if the binding and service definition for this Web service are held in separate WSDL files, then type the URL of the WSDL file that defines the binding).

If the WSDL is located through a UDDI registry, type `<i>uddiReference</i>`, `<i>serviceKey</i>` where

- *uddiReference* is the reference name by which a currently-deployed UDDI reference is known within the gateway (and by which it is listed using the **UDDI References -> List** option)
- *serviceKey* is the service key that the UDDI registry has assigned to the service. This is an example of a UDDI service key: 34280367-0ECF-46CE-B804-14C21D6D0FB1. For more information about UDDI service keys see *Publishing a Web service to a UDDI registry for deployment to the gateway*.

Note:

- When the Web services gateway deploys the Web service, it generates a matching "external" WSDL file that it makes available to gateway users. This "external" WSDL file also describes the service, but is located at a new URL and is generated and maintained by the Web services gateway itself.
- If the Web service is also published to one or more UDDI registries, then the "internal" WSDL file is required to remove the service from the gateway.

Location Type

Select the type of location (either URL or UDDI) where the "internal" WSDL file is held.

Target Service Name

If the Web service WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the target service's name from the target service WSDL.

Target Service Namespace

If the Web service WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the namespace of the target service's name from the target service WSDL.

Target Service Identity Information

Type the identity by which the Web service is known within the Web services gateway. This identity need not be unique.

- h. If you want to publish the service to one or more UDDI registries (selected in a previous step), enter the UDDI business key in the field provided under **UDDI Publication Properties**. This key identifies the business category under which you want your service to appear in UDDI. To get a list of valid business keys, look up businesses in a UDDI registry. This is an example of a UDDI business key: 08A536DC-3482-4E18-BFEC-2E2A23630526. For more information about UDDI business keys see Publishing a Web service to a UDDI registry for deployment to the gateway.

4. Click **OK**.

Results

If the processing completes successfully, the list of deployed Web services is updated to include the new Web service. Otherwise, an error message is displayed.

What to do next

If you enabled the check box 'Authorization Policy - Control access to this service', you must now enable Web service operation-level authorization.

Data type representation - choosing between Generic classes and Deployed Java classes

When you deploy a Web service, the **Message part representation** option allows you to choose between **Generic classes** and **Deployed Java classes**.

As your message passes through the gateway, the message parts are represented as actual Java objects. The data type used for each part is defined as follows:

- It is one of the set of XML schema and SOAP supported simple and compound data types, or
- It is a complex type defined in the WSDL schema section.

Note: A complex type is a data type represented by a Java class (such as a user-written class) that is not part of the native Java language.

Generic classes and **Deployed Java classes** can both represent simple, compound and complex data types, subject to the following constraints:

- The gateway only supports the simple and compound types that are listed in Web services gateway - Supported types.
- Only **Deployed Java classes** can represent Vectors, Enumerations, Hashtables and Maps (but either **Generic classes** or **Deployed Java classes** can represent Arrays).
- If **Deployed Java classes** are used to represent complex types, then the actual Java classes representing these complex types must be deployed to the application server on which the gateway is running.

- If the target service uses Java or EJB WSDL bindings (that is, if the target service is a Java class deployed on the local application server, or it is an enterprise bean) then **Deployed Java classes** must be used, and the bindings must be made available as described in Deploying Web services with Java bindings.

The gateway's schema parser determines all top-level types that are defined in the WSDL schema section, and generates mappings to **generic classes** for all of these types. This enables the gateway to forward requests (and responses) containing most complex data type parameters (and return values) to a remote destination without requiring the actual Java classes representing these complex types to be deployed to the application server on which the gateway is running. So if your Web service uses complex data types, and there is no other constraint that forces you to use **Deployed Java classes**, then you should select **Generic classes**.

Performance is the same whether you choose to use **Deployed Java classes** or **Generic classes**.

Complex data types - mapping namespaces to packages

If you write your own WSDL file that describes your Web service (rather than use an automated tool such as WebSphere Studio Application Developer) and the service uses complex data types, then to ensure that it can be successfully deployed to the gateway you should follow these guidelines for mapping namespaces to packages.

For working with complex data types, there is no industry-wide standard way of mapping namespaces to packages (in fact the JAX-RPC standard states that the tools themselves must make up their own standard). The standard used by the gateway is as follows:

- Set the namespace of the complex data type to the class's java package name.
- Set the complex data type name to the class's name.

So if the java class you are using for the complex data type is `random.RandomData`, then the namespace of the complex data type is `random`, and the complex data type name is `RandomData`. And if the full package name is `com.ibm.www.random`, then the namespace is `www.ibm.com/random`.

See also the troubleshooting tip about working with Web services that use complex data types.

Deploying Web services with Java bindings

For Web services deployed with Java bindings (or EJB bindings even where the Web service is on a different server) you must make additional classes available to the gateway.

For **EJB bindings**, make the EJB client JAR file available. If the Web service is deployed on the same server as the gateway, the necessary interfaces and classes are already visible. If not, you should implement one of the following options:

- Copy the EJB client JAR file into the `<i>WAS_HOME</i>/lib` or `<i>WAS_HOME</i>/lib/app` directory (where `WAS_HOME` is the root directory for your installation of IBM WebSphere Application Server).
- Update the application server class path to include the EJB client JAR file.

For **Java bindings**, make the Java classes for the Web service available by implementing one of the following options:

- Copy the JAR or class files that contain the Java classes into the `<i>WAS_HOME</i>/lib` or `<i>WAS_HOME</i>/lib/app` directory.
- Update the application server class path to include the JAR file.
- Wrap the Java classes in an enterprise bean and deploy it on the same application server. WebSphere Application Server will then make the classes available to the gateway application.

Web services gateway - Supported types

When you deploy a Web service, you use the **Message part representation** option to choose between Generic classes and Deployed Java classes. This topic gives reference information about the data types that are supported in each case.

Simple types

The following table gives a list of the XML schema (and Java equivalent) simple types that are supported by both **Generic classes** and **Deployed Java classes**:

XML schema (and Java equivalent) simple type

xsd:string
 xsd:float
 xsd:double
 xsd:int
 xsd:boolean
 xsd:byte
 xsd:short
 xsd:long
 xsd:decimal
 xsd:QName
 xsd:date
 xsd:timeInstant

Compound types

From a Java perspective, compound types are types with constituent elements. These elements are either identified purely by name (for example, a Java class with several member properties) or by ordinal position (for example, a List data structure like Array or Vector).

The following table gives a list of the SOAP (and Java equivalent) compound types that are supported:

SOAP compound type	java equivalent compound type	Supported by
Array	Java array	Generic classes and Deployed Java classes
Vector	java.util.Vector	Deployed Java classes
Vector	java.util.Enumeration	Deployed Java classes
Map	java.util.Hashtable	Deployed Java classes
Map	java.util.Map	Deployed Java classes

Complex types

A complex type is a data type represented by a Java class (such as a user-written class) that is not part of the native Java language.

Complex types can include combinations of simple types, compound types, and other complex types. For example, the children of a complex type might be represented by another complex type, or by any of the simple or compound types.

Generic classes and **Deployed Java classes** can both represent complex types, subject to the constraints described in Data type representation - choosing between Generic classes and Deployed Java classes.

Publishing a Web service to a UDDI registry for deployment to the gateway

The gateway interacts with UDDI registries as described in UDDI registries - Web service directories that integrate with the Web services gateway. When you deploy a Web service to the gateway, you enable UDDI interaction by entering a UDDI reference, and (depending upon what you are trying to do) either or both of the following pieces of information:

- The "service key" that the UDDI registry has assigned to this service.
- The "business key" that identifies the UDDI business category under which you want your service to appear in the UDDI registries.

You get these two keys *from the UDDI registry*. To help you understand what UDDI "service keys" and "business keys" are, and where you find them in a UDDI registry, this topic describes how to publish a Web service to a UDDI registry.

Note:

- The UDDI publication process described below requires that you specify a "technical model". Technical models are generic categories. They allow a UDDI registry user to search for a *type* of service, rather than needing to know the access details for a specific service. The gateway makes no use of technical model information, because it only interacts with UDDI registries at the level of specific Web services.
- The following task steps include specific navigation instructions. These instructions describe how you publish a Web service to the (IBM WebSphere UDDI Registry). If you are working with a different UDDI registry, then the specific navigation will be different but the underlying principles will be the same.

Steps for this task

1. Specify a business:
 - a. To get a list of valid business keys, look up businesses in the UDDI registry.
This is an example of a UDDI business key: 08A536DC-3482-4E18-BFEC-2E2A23630526.
 - b. If you do not find an appropriate existing business in the UDDI registry, then use the **Add a business** option on the **Advanced Publish** section of the Publish pane to add a new one.
2. Add a technical model:
 - a. Select **Add a technical model** on the **Advanced Publish** section of the Publish pane.

- b. Enter the name as specified for the target namespace of your binding (or interface) WSDL file, then add a description (if required).
 - c. Add a category of Type unspsc and value wsdlSpec (the Key name field can be left blank).
 - d. Add an overview URL specifying the URL for your binding WSDL file, then add a description (if required).

Note: The binding and the service definition for your Web service might be held in separate WSDL files, so be careful to type the URL of the WSDL file that defines the *binding*.
 - e. Click **Publish Technical Model**.
3. Add a service:
 - a. Select **Show owned entities** on the **Advanced Publish** section of the Publish pane.
 - b. Select **Add a Service** for your business.
 - c. Enter the name as specified for the target service in your WSDL file, then add a description (if required).
 - d. For the **Access point** ensure the correct URL type is selected (for example http for an http access point), then enter the value of the soap:address location (or its equivalent) from your service definition WSDL file (for example http://yourhost:80/SimpleTest/servlet/rpcrouter).
 - e. For the **Technical model** select **Add**, then find the required technical model by entering a suitable prefix and selecting **Find technical models**, then check the selection box for the required technical model and click **Update**.
 - f. Click **Publish Service**.

Results

The UDDI registry assigns a **service key** to your service, and publishes the service.

Removing Web services from the Web services gateway

To remove a Web service, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Services

- Remove

The main pane is updated with a list of all the Web services that are deployed to the Web services gateway. Alongside each entry in the list is a check box.

3. Select the check box for every Web service that you want to remove.
4. Click **OK**.

Results

If the processing completes successfully, the list of deployed Web services is updated. Otherwise, an error message is displayed.

Note: If the Web service that you want to remove is also published to one or more UDDI registries, then the "internal" WSDL file is required to remove the service from the gateway. So if the service is published to UDDI, and the processing does

not complete successfully, check that the WSDL file is still available at the location defined for the service in the **Target Services -> WSDL Location**. For more information see Listing and managing gateway-deployed Web services.

Running the Web services gateway samples

The following pre-built samples are available for use with the Web services gateway:

- The standard Stock Quote service sample, that requires an Internet connection.
- The Address Book service sample, that allows the storing and retrieval of names and addresses.

Results

These samples, and documentation on how to use them, are available through the Web services gateway samples link on the Samples Central page (<http://www.ibm.com/websphere/developer/library/samples/AppServer.html>) of the IBM WebSphere Developer Domain Web site.

What to do next

If you want to test the gateway taking service definitions from a private UDDI registry such as the (WebSphere UDDI Registry), you should complete the following additional steps:

1. Publish the WSDL for each of these samples to UDDI. (For more information on how to do this, see Publishing a Web service to a UDDI registry for deployment to the gateway and the documentation for your private UDDI registry).
2. Instruct the gateway to locate the service through the UDDI registry, as described in Deploying Web services to the Web services gateway.

Passing SOAP messages with attachments through the Web services gateway

The Web services gateway supports Web services that pass attachments in a MIME message. This support is included in the SOAP/HTTP channel.

Attachments are carried through the various gateway components and passed on to the target service. The content MIME type of each attachment is preserved.

When the Target service is deployed to a JAX-RPC compliant server, the attachments can be accessed on the target service using `javax.activation.DataHandler`.

The WSDL representing a SOAP messages with attachments service must define the attachment parts in the Binding section. The `mime:multipartRelated`, `mime:part` and `mime:content` tags are used to describe the attachment.

For more information, see the following topics:

- SOAP messages with attachments - a definition.
- Writing the WSDL extensions for SOAP messages with attachments.

Note: The following scenarios are not supported:

- Using the Apache SOAP channel.

- Using DIME.
- Using the mime:mimeXml WSDL tag.
- Nesting a mime:multipartRelated inside a mime:part.
- Using Arrays or Vectors of DataHandlers, Images, and so forth.

The MIME headers from the incoming message are not preserved for referenced attachments. The outgoing message contains new MIME headers for Content-Type, Content-Id and Content-Transfer-Encoding that are created by WSIF.

SOAP messages with attachments - a definition

From an architecture and external specification viewpoint, SOAP Messages with Attachments(<http://www.w3.org/TR/SOAP-attachments>) is an extension to the SOAP 1.1 (<http://www.w3.org/TR/SOAP/>) Recommendation from the World Wide Web Consortium (W3C)(<http://www.w3.org/>).

The W3C SOAP messages with attachments document describes a standard way to associate a SOAP message with one or more attachments in their native format (for example GIF or JPEG) by using a multipart MIME structure for transport. It defines specific usage of the Multipart/Related MIME media type, and rules for the usage of URI references to refer to entities bundled within the MIME package. It thereby outlines a technique for a SOAP 1.1 message to be carried within a MIME multipart/related message in such a way that the SOAP processing rules for a standard SOAP message are not changed.

An associated W3C document Web services Description Language (WSDL)(<http://www.w3.org/TR/wsdl>) outlines a technique for including bindings to MIME types in a WSDL file.

Writing the WSDL extensions for SOAP messages with attachments

Usage scenario

The following WSDL illustrates a simple operation that has one attachment called attch:

```
<binding name="MyBinding" type="tns:abc" >
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="MyOperation">
    <soap:operation soapAction=""/>
    <input>
      <mime:multipartRelated>
        <mime:part>
          <soap:body parts="part1 part2 ..." use="encoded"
            namespace="http://mynamespace"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
        </mime:part>
        <mime:part>
          <mime:content part="attch" type="text/html"/>
        </mime:part>
      </mime:multipartRelated>
    </input>
  </operation>
</binding>
```

Note:

- There must be a part (in this example attach) on the message for the operation (in this example MyOperation). There can be other input parts to MyOperation that are not attachments.
- In the binding input there must either be a <soap:body tag or a <mime:multipartRelated tag, but not both.
- For MIME messages, the soap:body is inside a mime:part. There may be one mime:part that contains a soap:body in the binding input and that must not contain a mime:content as well, because a content type of text/xml is assumed for the soap:body.
- There can be multiple attachments in a MIME message, each described by a mime:part.
- Each mime:part (that is not a soap:body) contains a mime:content that describes the attachment itself. The type attribute inside the mime:content is not checked or used by the gateway. It is there to suggest to the application using the gateway what the attachment contains. Multiple mime:contents inside a single mime:part means that the backend service will expect a single attachment with a type specified by one of the mime:contents inside that mime:part.
- The parts="..." attribute inside the soap:body is assumed to contain the names of all the SOAP parts in the message, but not the attachment parts. If there are only attachment parts, then you must specify parts="" (empty string). If you omit the parts attribute altogether, then the gateway assumes ALL parts including the attachments - which means the attachments will appear twice.

In your WSDL you might have defined a schema for the attachment (for instance as a binary[]). Whether or not you have done this, the gateway silently ignores this mapping and treats the attachment as a Data Handler.

Unreferenced attachments need not be mentioned in the WSDL bindings at all.

Administering security for the Web services gateway

The gateway provides basic authentication and authorization facilities based upon the broader security features of WebSphere Application Server. See [Enabling basic authentication and authorization for the gateway](#).

The gateway can also invoke Web services that include https:// in their addresses, if you have configured your Java and WebSphere Application Server security properties to allow it. To check your security property settings, see [Invoking Web services over HTTPS](#).

What to do next

For hints on solving security-related problems, see "Web services gateway troubleshooting tips".

Enabling Web Services Security (WS-Security) for the gateway

You can configure the gateway for secure transmission of SOAP messages using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) draft recommendation. For more information on how WS-Security is implemented in WebSphere Application Server Network Deployment, see [Securing Web services](#). For more information on the approach taken by the gateway to implementing this emerging standard, see [The Web services gateway and WS-Security](#).

The gateway sits between the service requester (the client) and the target Web service. You configure the gateway to act as the target service from the point of view of the client, and as the client from the point of view of the target service. So you need to get, from the owning parties, the WS-Security configurations for both the client and the Web service. This information is found in the following files on the owners systems:

- Key stores (.ks and .jceks files).
- Certificate stores (.cer files).
- Security settings (ibm-webservicesclient-ext.xmi for the client, and ibm-webservices-ext.xmi for the Web service).
- Binding information - for example the location of a keystore on the file system (ibm-webservicesclient-bnd.xmi for the client, and ibm-webservices-bnd.xmi for the Web service).

Note: If the client is hosted on WebSphere Application Server, and the Web service security settings were created using IBM Web services tooling (for example WebSphere Studio Application Developer), then the files that contain the security settings and binding information will have the exact file names (*.xmi) given above. For clients and Web services from other vendors, these files will have different names.

You need to copy the key store and certificate store files to the gateway file system, and to enter and configure for the gateway the security settings that are contained in the .xmi files. The security settings are entered and configured manually using the gateway administrative user interface. There are tools available (for example WebSphere Studio Application Developer) that can parse the .xmi files for you.

You use the **Gateway -> Security** option to configure the security bindings (the tokens, keys, signatures and encryption methods) that are available to the gateway, as described in Configuring the gateway security bindings.

You then configure the level of security that applies at each stage of the transmission (and note that different levels of security, including no security, can be applied to each stage):

- From the service requester to the gateway.
- From the gateway to the target service.
- From the target service back to the gateway.
- From the gateway back to the service requester.

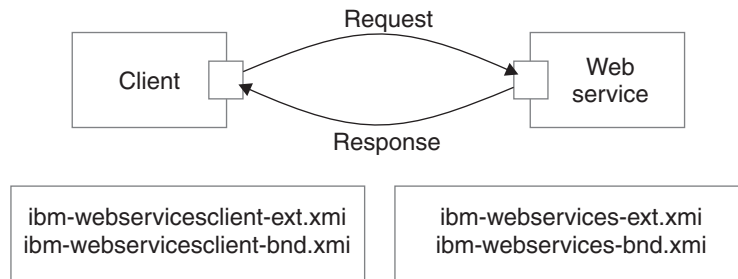
For information on how to do this, see the following topics:

- Editing the service security configuration - how to configure secure communication for this gateway service between the service requester (the client) and the gateway.
- Editing the target service security configuration - how to configure secure communication between the gateway and the target service.

The Web services gateway and WS-Security

You can configure the gateway for secure transmission of SOAP messages using tokens, keys, signatures and encryption in accordance with the emerging Web Services Security (WS-Security) specification(<http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>).

In a normal (non gateway) WS-Security scenario, the flow is as shown in the following figure:

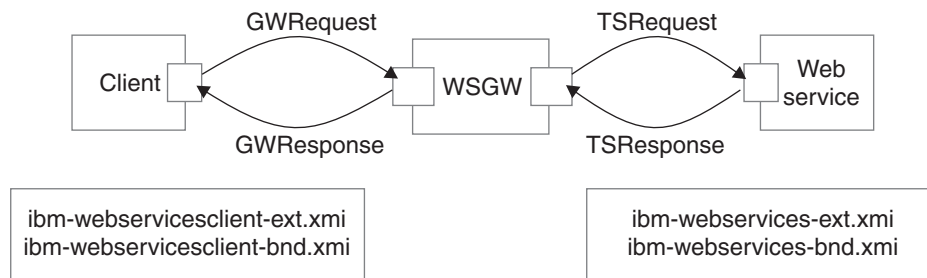


The client generates a request which is then handled by the client Web services engine. It reads the client security configuration and applies the security defined in the `ibm-webservicesclient-ext.xmi` file to the SOAP message. It gets additional binding information from the `ibm-webservicesclient-bnd.xmi` file (for instance, the location of a keystore on the file system).

On receipt of a SOAP message, the Web services engine on the server refers to the `*.xmi` files for the called Web service. In this case, the `ibm-webservices-ext.xmi` file tells the engine what security the incoming message must have (for example, that the body of the message must be signed). If the message does not comply, then it is rejected. The Web services engine verifies any security information, then passes the message on to the Web service being called.

On the response leg from server to client, the process is reversed. The Web service `*.xmi` files tell the Web services engine what security to apply to the response message, and the client `*.xmi` files tell the client engine what security to require in the response message.

When the gateway is introduced, the scenario is more complex. Essentially it can be thought of as two separate request/response invocations. Client to gateway and gateway to target service, as shown in the following figure:



In this scenario, the client and the Web service are unchanged, and still have the same security settings in their `*.xmi` files. However, the gateway is unsecured. Secure SOAP messages cannot travel through the gateway unchanged, and must be processed on receipt. So the gateway needs to act as the target service from the point of view of the client, and as the client from the point of view of the target service. This scenario means that the security settings for the Web service need to be configured for the view of the service that the gateway presents to the client, and the security settings for the associated gateway target services (remember that

there may be multiple target services deployed for a single gateway service) need to be configured with the security settings for the client.

WS-Security settings for the gateway are configured manually using the gateway administrative user interface.

Configuring the gateway security bindings

You use the **Gateway -> Security** option to configure all the security bindings (the tokens, keys, signatures and encryption methods) that are available to the gateway. This information describes the security bindings that are used to secure the SOAP messages that pass between service requesters (clients) and the gateway, and between target services and the gateway.

You receive this security binding information direct from the service requester or target service provider, in the form of an `ibm-webservicesclient-bnd.xmi` file for the client, and an `ibm-webservices-bnd.xmi` file for the Web service. You extract the information from these `.xmi` files, then manually enter it into the gateway security bindings forms that are described below.

Steps for this task

1. Display the Web services gateway administrative user interface.
2. In the navigation pane, click the following link:

Gateway

- Security

The **Configure Gateway Security Bindings** form is displayed.

This form is divided into eight sections, one for each type of security binding. The following comments apply to every section:

- To add a new binding, click **Add new *binding_type***.
- To amend an existing binding, click on the name of the binding.
- To delete an existing binding, click **remove** alongside the name of the binding.
- If you choose to add or amend a binding, then a binding information sub-form is displayed. Within this form:
 - Help is provided in comments on the sub-form, and in hover-help alongside each field.
 - Fields marked on the sub-form with an asterisk (*) are required fields.
 - For some bindings, the fields that are required are different depending on whether you are *adding* or *amending* a binding.
 - Many fields are populated by making a selection from a drop-down list.
 - Information entered in one binding information sub-form often appears in a drop-down list in another sub-form.
 - When you have finished entering information for a binding, click **OK**.

3. Add, amend or delete **Signing Information**.

The *signing information* specifies the configuration for digital signature validation and message signing.

If you choose to add or amend a signing information binding, then the **Define Signing Information** form is displayed. In this form you enter a name for the binding (if you are adding a new binding), and the following binding details:

- Signature method

- Digest method
- Canonicalization method
- Signing key name
- Signing Key Locator
- Certificate Path (a choice between trusting *any* certificates, and trusting certificates from a particular *Trust Anchor* and *Certificate Store*).

Note: The signing information can also be used for X.509 certificate validation when **Authentication Method** is IDAssertion and **ID Type** is X509Certificate in the gateway service configuration. In such cases, you must fill in the "Certificate Path" fields only.

4. Add, amend or delete **Encryption Information**.

The *encryption information* specifies the configuration for encrypting and decrypting messages.

If you choose to add or amend an encryption information binding, then the **Define Encryption Information** form is displayed. In this form you enter a name for the binding (if you are adding a new binding), and the following binding details:

- Key locator
- Encryption key name
- Key encryption algorithm
- Data encryption algorithm

5. Add, amend or delete **Trust Anchors**.

A *trust anchor* specifies a list of key store configurations that contain root trusted certificates.

If you choose to add or amend a trust anchor, then the **Define Trust Anchor** form is displayed. In this form you enter a name for the trust anchor (if you are adding a new anchor), and the following details:

- Key store type
- Key store path
- Key store password

Note: These configurations are used for certificate path validation of the incoming X.509-formatted security tokens. The keystore must be created using the Java Development Kit keytool. The ikeyman tool is not supported.

6. Add, amend or delete **Certificate Stores**.

A *certificate store* specifies a list of untrusted, intermediate certificate files. It is used for certificate path validation of incoming X.509-formatted security tokens.

If you choose to add or amend a certificate store, then the **Define Certificate Store** form is displayed. In this form you enter a name for the certificate store (if you are adding a new store), and the following details:

- Certificate Store Provider

When you amend an existing certificate store, you are given an extra option to add or remove X.509 certificates from the list of certificates that are contained within this store. When you add an X.509 certificate, you specify the full path for the certificate.

7. Add, amend or delete **Key Locators**.

A *key locator* specifies a configuration that is used to retrieve keys for signature and encryption. A key locator class can be customized to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

If you choose to add or amend a key locator, then the **Define Key Locators** form is displayed. In this form you enter a name for the key locator (if you are adding a new key locator), and the following details:

- Classname
- Key store type
- Key store path
- Key store password

When you amend an existing key locator, you are given two extra options:

- Add or remove key entries. For each additional key, you specify:
 - Key name
 - Key alias
 - Key password

Note: You do not need to list all the certificate entries as keys; instead, the distinguished name (DN) of the certificate is used as the search key.

- Add or remove additional properties for the configuration. For each additional property, you specify:
 - Property name
 - Property value

8. Add, amend or delete **Trusted ID Evaluators**.

A *trusted ID evaluator* determines whether the identity (ID)-asserting authority is trusted.

If you choose to add or amend a trusted ID evaluator, then the **Define Trusted ID Evaluator** form is displayed. In this form you enter a name for the trusted ID evaluator (if you are adding a new evaluator), and the following details:

- Class name

When you amend an existing trusted ID evaluator, you are given an extra option to add or remove additional properties for the configuration. For each additional property, you specify:

- Property name
- Property value

9. Add, amend or delete **Login Mappings**.

A *login mapping* specifies a configuration for validating security tokens within incoming messages.

If you choose to add or amend a login mapping, then the **Define Login Mapping** form is displayed. In this form you enter a name for the login mapping (if you are adding a new mapping), and the following details:

- Authentication method
- JAAS Configuration name
- Callback handler factory class name
- Token type local name
- Token type URI

When you amend an existing login mapping, you are given extra options to add or remove additional properties for the *configuration*, and additional properties for the *callback handler factory*. For each additional property, you specify:

- Property name
- Property value

10. Add, amend or delete **Login Bindings**.

A *login binding* specifies a configuration for generating security tokens within outgoing messages.

If you choose to add or amend a login binding, then the **Define Login Binding** form is displayed. In this form you enter a name for the login binding (if you are adding a new binding), and the following details:

- Authentication method
- JAAS Configuration name
- Callback handler factory class name
- Token type local name
- Token type URI

When you amend an existing login binding, you are given extra options to add or remove additional properties for the *configuration*, and additional properties for the *callback handler*. For each additional property, you specify:

- Property name
- Property value

Editing the service security configuration

Before you begin

Before you can select the security settings that are applied for an individual Web service, you must configure the gateway security bindings.

For each Web service, you can select the security settings that are applied between the service requester (the client) and the gateway. These settings are specified for each stage of the transmission:

- From the service requester to the gateway (the client request).
- From the gateway back to the service requester (the gateway response).

You receive this security settings information from the service requester and from the target service provider in the following form:

- An `ibm-webservicesclient-bnd.xml` for the client, and `ibm-webservices-bnd.xml` for the Web service, from which you process the security bindings information as described in *Configuring the gateway security bindings*.
- An `ibm-webservicesclient-ext.xml` for the client, and `ibm-webservices-ext.xml` for the Web service, which contain the information on the levels of security (integrity, confidentiality and identification) that are required when this Web service exchanges messages with a service requester. These are therefore also the settings that the gateway needs to apply when it makes the equivalent gateway service available to a service requester.

To set the security settings that are applied between the service requester (the client) and the gateway, complete the following steps:

Steps for this task

1. List the gateway-deployed Web services
2. Click the name of a Web service in the list.

A form is displayed through which you can view and modify the current deployment details for this Web service, and add or remove multiple target services.

3. In the **Service Security** section, select the **Edit service security configuration** option.

The service security configuration form is displayed. This form is divided into the following sections:

- **Gateway Security Properties** (the Actor URI)
- **Client Request Security Properties** (integrity, confidentiality and identification settings)
- **Gateway Response Security Properties** (the response Actor URI, and integrity and confidentiality settings)
- **Security bindings** (request bindings and response bindings).

The following comments apply to every section:

- Help is provided in comments on the form, and in hover-help alongside each field.
 - There are no required fields.
 - Many fields are populated by making a selection from a drop-down list.
4. In the **Gateway Security Properties** section, set the Actor URI.
Note: If you specify an Actor URI, then only SOAP security headers with this Actor URI will be processed.
 5. In the **Client Request Security Properties** section, set the following security levels:
 - a. Set the **Integrity** level.
Set the parts of the incoming SOAP message that must be signed (the Body, the Timestamp and the Security Token).
 - b. Set the **Confidentiality** level.
Set the parts of the incoming SOAP message that must be encrypted (the Body and the Username Token).
 - c. Set the **Identification** level.
Set the identification methods that will be accepted (Basic Authentication , Digital Signature and ID Assertion).
 6. In the **Gateway Response Security Properties** section, set the following security levels:
 - a. Set the Response Actor URI.
Note: If you specify a Response Actor URI, then the SOAP security header in the response message will have this Actor URI.
 - b. Set the **Integrity** level.
Set the parts of the response SOAP message that must be signed (the Body and the Timestamp).
 - c. Set the **Confidentiality** level.
Set whether or not the Body of the response SOAP message must be encrypted.
 7. In the **Security bindings** section, set the Request bindings and the Response bindings that are to be used.

Note: You choose these bindings (Signing Information, Encryption Information, Trusted ID Evaluator and Login Mappings) from pull-down lists. The available items in these lists are those that you have previously entered as described in Configuring the gateway security bindings.

8. When you have finished editing the service security configuration, click **Apply changes**.

Editing the target service security configuration

Before you begin

Before you can select the security settings that are applied for target Web service, you must configure the gateway security bindings.

For each target Web service, you can select the security settings that are applied between the target Web service and the gateway. These settings are specified for each stage of the transmission:

- From the gateway to the target service (the target service request).
- From the target service back to the gateway (the target service response).

You receive this security settings information from the service requester and from the target service provider - usually in the following form:

- An `ibm-webservicesclient-bnd.xmi` for the client, and `ibm-webservices-bnd.xmi` for the Web service, from which you process the security bindings information as described in Configuring the gateway security bindings.
- An `ibm-webservicesclient-ext.xmi` for the client, and `ibm-webservices-ext.xmi` for the Web service, which contains the information on the levels of security (integrity, confidentiality and identification) that are required when this Web service exchanges messages with a service requester. These are therefore also the settings that the gateway needs to apply when it calls the target service on behalf of the service requester.

To set the security settings that are applied between the target service and the gateway, complete the following steps:

Steps for this task

1. List the gateway-deployed Web services
2. Click the name of a Web service in the list.
A form is displayed through which you can view and modify the current deployment details for this Web service, and add or remove multiple target services.
3. In the **Target Services** section, click the name of a target Web service in the list.
A form is displayed, containing the same fields that you filled in when you added the target service, and also the following additional fields:
 - **Started** (a check box).
 - **Enable target service security** (a check box).
 - **Edit target service security configuration**.
4. Select the **Edit target service security configuration** option.
The target service security configuration form is displayed. This form is divided into the following sections:
 - **Target Service Security Properties** (the Actor URI)

- **Target Service Request Security Properties** (integrity, confidentiality and identification settings)
- **Target Service Response Security Properties** (the response Actor URI, and integrity and confidentiality settings)
- **Security bindings** (request bindings and response bindings).

The following comments apply to every section:

- Help is provided in comments on the form, and in hover-help alongside each field.
 - There are no required fields.
 - Many fields are populated by making a selection from a drop-down list.
5. In the **Target Service Security Properties** section, set the Actor URI.
Note: If you specify an Actor URI, then only SOAP security headers with this Actor URI will be processed.
 6. In the **Target Service Request Security Properties** section, set the following security levels:
 - a. Set the Target Actor URI.
Note: If you specify a Target Actor URI, then the SOAP security header in the request message will have this Actor URI.
 - b. Set the **Integrity** level.
Set the parts of the outgoing SOAP message that must be signed (the Body, the Timestamp and the Security Token).
 - c. Set the **Confidentiality** level.
Set the parts of the outgoing SOAP message that must be encrypted (the Body and the Username Token).
 - d. Set the **Identification** level.
Set the identification methods that will be accepted (Basic Authentication , Digital Signature and ID Assertion).
 7. In the **Target Service Response Security Properties** section, set the following security levels:
 - a. Set the **Integrity** level.
Set the parts of the response SOAP message that must be signed (the Body and the Timestamp).
 - b. Set the **Confidentiality** level.
Set whether or not the Body of the response SOAP message must be encrypted.
 8. In the **Security bindings** section, set the Request bindings and the Response bindings that are to be used.
Note: You choose these bindings (Signing Information, Encryption Information, Trusted ID Evaluator and Login Mappings) from pull-down lists. The available items in these lists are those that you have previously entered as described in Configuring the gateway security bindings.
 9. When you have finished editing the target service security configuration, click **Apply changes**.

Enabling basic authentication and authorization for the gateway

Basic authentication can be applied at two levels, as described in the following topics:

1. Enabling gateway-level authentication.
2. Enabling Web service operation-level authorization.

For **gateway-level authentication**, you set up a role and realm for the gateway on WebSphere Application Server's Web server and servlet container, and define the userid and password that is used by the gateway to access the role and realm. You also modify the gateway's channel applications so that they only give access to the gateway to service requesters that supply the correct userid and password for that role and realm.

Note: This means that gateway-level authentication must be enabled before you install any channels.

For **operation-level authorization**, you apply security to individual methods in a Web service. To do this, you create an enterprise bean with methods matching the Web service operations. These EJB methods perform no operation and are just entities for applying security. Existing WebSphere Application Server authentication mechanisms can be applied to the enterprise bean. Before any Web service operation is invoked, a call is made to the EJB method. If authorization is granted, the Web service is invoked. Your target Web service is protected by wrapping it in an EAR file, and applying role-based authorization to the EAR file. This process is explained in general terms in Operation-level security - role-based authorization.

Note:

- If you want to enable operation-level authorization, you must first enable gateway-level authentication.
- If you want to change the default gateway-level authentication settings, you must do so before you install any channels.
- After gateway-level authentication has been enabled, filters have access to the requester's authentication information.

The Web services gateway can also invoke Web services that include https:// in their addresses, if the Java and WebSphere security properties have been configured to allow it. To check your security property settings, see the following topic:

- [Invoking Web services over HTTPS](#)

[What to do next](#)

For hints on solving security-related problems, see "Web services gateway troubleshooting tips".

Enabling gateway-level authentication

A number of default gateway-level authentication settings are included in the gateway. There is a default role of AuthenticatedUsers which includes the special group 'AllAuthenticatedUsers'. When security is enabled, you must supply a user ID and password to use the gateway administrative interface or invoke a gateway service.

This task covers the three main areas in which you might want to make changes:

- Changing the default gateway-level authentication settings.
- Enabling gateway-level authentication.
- Assigning users and groups to roles.

Note:

- If you want to change the default gateway-level authentication settings, you must do so before you install any channels. When you run the script that installs the gateway itself (*either* into a deployment manager cell *or* into a stand-alone application server) you also install the following channels:

- Apache SOAP channel 1.

So if you change the default gateway-level authentication settings after you install the gateway, you then need to re-run the gateway install.

- You can enable gateway-level authentication, and assign users and groups to roles, at any time.
- After gateway-level authentication has been enabled, filters have access to the requester's authentication information.

Steps for this task

1. To change the default gateway-level authentication settings, use the WebSphere Application Server Application Assembly Tool (AAT) to complete the following steps:
 - a. Set up a role and realm for the gateway on WebSphere Application Server's Web server and servlet container.
 - b. Define the user ID and password that is used by the gateway to access the role and realm.
 - c. Modify the gateway's channel applications so that they only give access to the gateway to service requesters that supply the correct user ID and password for that role and realm.
2. To enable gateway-level authentication, complete the following steps:
 - a. Start the WebSphere Application Server administrative server.
 - b. Start the administrative console.
 - c. In the navigation pane, select **Security -> Global Security**.
 - d. In the main pane, on the **Configuration** tab, enable the "Enabled" check box.
 - e. Save the settings.
 - f. Stop then restart the application server.
 - g. Close the administrative console.
3. You can use the AAT or the administrative console to assign users and groups to roles. To map users to roles using the administrative console, complete the following steps:
 - a. Start the WebSphere Application Server administrative server.
 - b. Start the administrative console.
 - c. In the navigation pane, select **Application -> Enterprise Applications -> wsgw**.
In the main pane, an option to map security roles to users and groups appears in the Additional Properties table.
 - d. Modify the security roles and save the settings.

- e. Repeat the previous two steps for each enterprise application that you want to modify.
- f. Stop then restart the application server.
- g. Close the administrative console.

For more information see Assigning users and groups to roles.

Note: The current jacl install scripts do not let you assign users to roles as part of installing the gateway into a deployment manager cell or into a stand-alone application server.

What to do next

You might now want to enable operation-level authorization, or install the gateway.

Enabling operation-level authorization

Use this task to apply security to individual methods in a Web Service.

Before you begin

Before you begin this task you must first enable gateway-level authentication.

You can only apply operation-level authorization to a Web service that has already been deployed to the gateway with the check box 'Authorization Policy - Control access to this service' enabled.

This task involves making changes to the file `/lib/wsgwauth.ear`. To protect the installation version of this file, you should make a backup copy of it before you change it.

For operation-level authorization you create an enterprise bean with methods matching the Web service operations. These EJB methods perform no operation and are just entities for applying security. Existing WebSphere Application Server authentication mechanisms can be applied to the enterprise bean. Before any Web service operation is invoked, a call is made to the EJB method. If authorization is granted, the Web service is invoked.

Your target Web service is protected by wrapping it in an EAR file, and applying role-based authorization to the EAR file. This process is explained in general terms in Operation-level security - role-based authorization.

The EAR file that now contains your Web service is then imported into `wsgwauth.ear` (which contains all of the gateway's protected Web services) and `wsgwauth.ear` is modified to set the roles and assign them to methods. Finally, this modified `wsgwauth.ear` file is deployed in Websphere Application Server and users are assigned to the previously defined roles.

To enable Web service operation-level authorization, complete the following steps:

Steps for this task

1. To create `<i>your_webservice</i>.ear`, complete the following steps:
 - a. Open a command prompt.
 - b. Go to directory `/WSGW/scripts/auth`

c. Enter the command `WSGWAAuthGen <i>location</i> <i>your_webservice</i>` where

- *location* is the URL for the gateway (this must include the root context)
- *your_webservice* is the name of the service as deployed in the gateway (this is case-sensitive)

For example `WSGWAAuthGen http://<i>host</i>:<i>port</i>/wsgwAddressBook` where *host* and *port* are the host name and port number for the application server on which the gateway is installed.

Note: The Web service name and operation name can contain characters (such as "-", ".", "&") that are disallowed in an EJB class name and method name. So these are translated during the generation process of `<i>your_webservice</i>.ear`. A message appears informing you of the name change.

`<i>your_webservice</i>.ear` is created in directory `/WSGW/scripts`. There is also a temporary directory `/WSGW/scripts/ejb`, which you can delete.

2. To finish assigning roles and protecting methods, use the WebSphere Application Server Application Assembly Tool (AAT) to complete the following steps:
 - a. Start the AAT.
 - b. From the File menu select **File > Open**, and browse to select file `/lib/wsgwauth.ear`.
 - c. To import `<i>your_webservice</i>.ear` into `wsgwauth.ear`, complete the following steps:
 - In the navigation pane, open the pop-up menu for **EJB Modules** and select **Import**
 - Browse to select file `/WSGW/scripts/<i>your_webservice</i>.ear`. The Select modules to import window opens.
 - In the Select modules to import window, select *your_webservice* and click **OK**.
 - The Confirm values window opens. Click **OK**.
 - In the navigation pane, expand **EJB Modules** to confirm that `<i>your_webservice</i>.ear` has been imported.
 - d. In the navigation pane, expand **EJB Modules > your_webservice.ear** and select **Security Roles**.
 - e. For every security role that you want to create, repeat the following steps:
 - From the pop-up menu for **Security Roles**, select **New**.
 - Type the name and description of the new security role, and click **OK**.
 - f. In the navigation pane, expand **EJB Modules > your_webservice.ear** and select **Method Permissions**.
 - g. For every defined role that you want to assign to a Web service method, repeat the following steps:
 - From the pop-up menu for **Method Permissions**, select **New**. The New Method Permission window opens.
 - Type the name of the new method permission, and click **ADD** for Methods. The Add Methods window opens.
 - In the Add Methods window, expand the tree for remote methods and select the method to be protected. Click **OK**. The Add Methods window closes.

- In the New Method Permission window, click **ADD** for Roles. Select a previously defined role from the list then click **OK**.
- h. To ensure that the authorization enterprise bean can reference the newly-imported enterprise bean, complete the following steps:
 - In the navigation pane, expand **WSGW Authorization group > Session Beans > Authorization** and select **EJB References**.
 - From the pop-up menu for **EJB References**, select **New**. The New EJB Reference window opens.
 - In the New EJB Reference window, on the **General** tab, type a name for the reference then use the 'Link' combination box to select the newly-imported EJB (all the other fields on this tab are populated automatically).
 - In the New EJB Reference window, on the **Bindings** tab, type the JNDI name as it appears in the bindings tab of the service enterprise bean (this should be in the form `websphere/WSGW/Security/<i>your_webservice</i>`).
 - Click **OK**. The New EJB Reference window closes.
 - i. From the AAT File menu, select **File > Generate Code For Deployment**.
 - j. Make a note of the name of the modified ear file, then click **Generate Now**.
 - k. From the AAT File menu, select **File > Save** to save the modified copy of `wsgwauth.ear`.
 - l. Close the AAT.
3. To install the modified copy of `Deployed_wsgwauth.ear`, complete the following steps:
 - a. Start the WebSphere Application Server Administrative Console.
 - b. In the navigation pane, select **Applications > Install an Application**.
 - c. Use **Install New Application** to install `Deployed_wsgwauth.ear`. Select the users or groups to be assigned to the roles when prompted.

Operation-level security - role-based authorization

During construction of an EAR file, roles can be defined and applied to methods. At deployment of the EAR file, individual users or groups can be assigned to roles. So you can use this feature of EAR files to add role-based security to your Web service.

For example: You have a Web service that controls access to important information, and you want to give read-only access to some users, and write access to others. So when you build the EAR file you define two roles READ and WRITE, then you apply the READ role to the `getData` method and the WRITE role to the `writeData` method. When you deploy the EAR file in WebSphere Application Server, you assign 'All Authenticated Users' to the READ role and individual users to the WRITE role. When a user tries to access `WebService.getData`, their user name and password is checked by the operating system or by [].

Invoking Web services over HTTPS

The Web services gateway can invoke Web services that include `https://` in their addresses, if the Java and WebSphere security properties have been configured to allow it. This means that one gateway can send a SOAP/HTTPS message direct to another gateway, rather than having to export services and have clients invoke them using HTTPS.

To enable your gateway to send and receive SOAP/HTTPS messages, confirm that your Java and WebSphere security properties are configured as described in the following steps:

Steps for this task

1. Check that there is a copy of file `ibmjsse.jar` in directory `<i>WAS_HOME</i>/java/jre/lib/ext` (where `WAS_HOME` is the root directory for your installation of IBM WebSphere Application Server).

2. Edit the security properties file `<i>WAS_HOME</i>/java/jre/lib/security/java.security` so that it includes entries for both the Sun security provider and the IBM security provider. For example:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.jsse.IBMJSSEProvider
```

Note: The order is significant. The Sun security provider must come before the IBM provider.

3. Use the WebSphere Application Server Administrative Console to set up the following equivalent system properties:

```
// truststore location
System.setProperty("javax.net.ssl.trustStore",
    "your_truststore_root_directory/TestSSL/key.jks");
// set truststore password
System.setProperty("javax.net.ssl.trustStorePassword",
    "your_truststore_password");
//use ibm reference implementation
System.setProperty("java.protocol.handler.pkgs",
    "com.ibm.net.ssl.internal.www.protocol");
```

Enabling proxy authentication for the gateway

The gateway requires access to the Internet for invoking Web services and for retrieval of WSDL files. Many enterprise installations use a proxy server in support of Internet routing, and many proxy servers require authentication before they grant access to the Internet. This requirement is supported in HTTP messaging by a "Proxy-Authorization" message header that contains encoded username and password credentials.

For messages passing through the gateway, you can enable and disable proxy authentication, and specify whether the authentication credentials are supplied by the service requester or by the gateway. If you specify *requester-supplied* credentials, the credentials in the HTTP message that the gateway receives are re-instantiated by the gateway in the equivalent message that it sends on to the proxy. If you specify *gateway-supplied* credentials, the gateway ignores any credentials in the incoming HTTP message and supplies its own credentials in the equivalent message that it sends on to the proxy.

Note: In certain circumstances, the gateway also creates and sends its own messages (for example for WSDL retrieval). In these cases, the gateway always supplies its own credentials to the authenticating proxy. So even if you enable proxy authentication and specify *requester-supplied* credentials, you must still supply credentials for the gateway.

To enable proxy authentication for the gateway, complete the following steps:

Steps for this task

1. Display the Web services gateway administrative user interface.

2. In the navigation pane, click the following link:

Gateway

- [Configure](#)

The gateway configuration form is displayed:

The screenshot shows the 'Configure Gateway' form in the IBM Web Services Gateway. The left navigation pane has 'Gateway Configure' selected. The main area contains two text input fields: 'Namespace URI for services' with the value 'urn:ibmwsgw' and 'WSDL URI for exported definitions' with the value 'http://PHJ2:9080/wsgw'. An 'Apply Changes' button is at the bottom.

3. Enable the **Enable proxy authentication** check box.
4. In the **Proxy user** field, type the proxy username for the gateway itself.
Note: If you enable proxy authentication then this field is compulsory, even if you also specify *requester-supplied* credentials as described below.
5. In the **Proxy password** field, type the associated proxy password for the gateway itself.
Note: If you enable proxy authentication then this field is compulsory, even if you also specify *requester-supplied* credentials as described in the next step.
6. To set the **Use Gateway proxy credentials for invoking WebServices** check box, complete one of the following two steps:
 - a. If you want to use *requester-supplied* credentials, then clear the **Use Gateway proxy credentials** check box.
With this setting, each incoming message to the gateway from a service requester is expected to contain a valid "Proxy-Authorization" HTTP message header. This header is re-instantiated by the gateway in the equivalent message that it sends on to the proxy.
Note: For gateway-initiated messaging, such as WSDL retrieval, the gateway supplies its own credentials in the HTTP messages that it sends to the proxy.
 - b. If you want to use *gateway-supplied* credentials, then enable the **Use Gateway proxy credentials** check box.
With this setting, a trust association is established between the gateway and the authenticating proxy. The gateway supplies its own credentials in all messages that it sends to the proxy, and no username or password is required from service requesters for invoking Web services.
7. Click **Apply Changes**.

8. You also need to provide the application server in which your gateway is running with machine details for the authenticating proxy and for any internal machines that do not require authentication. You do this by setting system properties in the WebSphere Application Server Java Virtual Machine as follows:
 - a. Start the WebSphere Application Server administrative server.
 - b. Start the administrative console.
 - c. In the navigation pane, select **Application Servers -> your_server_name -> Process Definition -> Java Virtual Machine -> Custom Properties**.
 - d. Set the following properties:
 - **http.proxySet** - Set this to true to tell the application server that it is required to work with an authenticating proxy.
 - **http.proxyHost** - Set this to the machine name of the authenticating proxy.
 - **http.proxyPort** - Set this to the port through which the authenticating proxy is accessed. For example 8080
 - **http.nonProxyHosts** - List the internal machines for which authentication is not required for routing through the proxy. Separate each machine name in the list with a vertical bar "|".

Note: This list must include the machine on which the gateway is installed.
 - e. Save the settings.
 - f. Stop then restart the application server.
 - g. Close the administrative console.

What to do next

Note: You also use the gateway configuration form to set the namespace URI and WSDL URI for the Web services gateway.

Web services gateway troubleshooting tips

This topic provides hints to help you resolve problems you experience when using the Web services gateway.

For information on resolving WebSphere-level problems, see Diagnosing and fixing problems.

To identify and resolve gateway-related problems, you can use the standard WebSphere Application Server trace and logging facilities. To enable trace for the gateway, set the application server's trace string to `com.ibm.wsgw.*=all=enabled`. If you encounter a problem that you think might be related to the gateway, you can check for error messages in the WebSphere Application Server administrative console, and in the application server's `stdout.log` file. You can also enable the application server debug trace to provide a detailed exception dump.

The gateway's user interface uses cascading style sheets to lay out its pages, and javascript to monitor progress and advise you as you fill in each on-screen form. So your Web browser must support javascript and cascading style sheets, and it must be configured so that javascript and style sheets are enabled. How you do this depends on which browser you use. For example for Netscape, you select **Edit -> Preferences**, click Advanced in the Category pane, then confirm that the **Enable Javascript** and **Enable style sheets** check boxes are selected.

A list of the gateway runtime system messages, with details of what each message means, is given in Web services gateway messages.

Here is a checklist of common problems:

You have managed to deploy your Web service in the Web services gateway but you are getting a class cast exception when you invoke the operation which takes an integer parameter.

Check that your client is using the version of soap.jar that is supplied in the WebSphere Application Server's /AppServer/lib/app directory. If you enable trace, you may see in the trace for the request <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema">

The gateway expects the 2001 version of the XML schema. Older versions of soap.jar (including 2.2) generate 1999 schema. If you have the soap.jar that is supplied with WebSphere Application Server in the client's class path, you should see 2001 schema in the request, which should then work.

The persistent state of the Web services gateway has become out of sync with the channel applications.

This can happen if you remove and reinstall the Apache SOAP applications. If you need to do this, then either ensure that all corresponding channels configured with the Web services gateway are removed, or remove and reinstall the Web services gateway at the same time.

Note: The Web services gateway application (wsgw.ear) must be installed before channel and filter applications. If the gateway application needs to be reinstalled, all channels and filters must be uninstalled first, then reinstalled after the gateway application.

You are getting SOAP fault messages, but cannot determine the precise problem from the fault message.

If you receive a SOAP fault message with a faultstring which is just the value of one of the parameters of the invocation, that means that that parameter's value was invalid. For example if you have a service which expects an int parameter and you send it a message containing the value "1.1", then the fault message you receive simply contains 1.1 as the fault string:

```
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>1.1</faultstring>
```

Note: This is Apache SOAP behavior, and not something that the gateway can do anything about.

If you receive a SOAP fault message containing an element that is not present in the WSDL for the target service, then the error message thrown can be difficult to identify. There are two possible scenarios:

- The WSDL is deployed to use **Generic Classes**. In this case the returned SOAP message contains an IllegalArgumentException exception. For example:

```
[Attributes={}] [faultCode=SOAP-ENV:Server]
[faultString=com.ibm.wsgw.WSGWException:
WSGW0043E: Exception while executing operation createEntry
service ExchangeService.
Exception: org.apache.wsif.WSIFException:
SOAPException: SOAP-ENV:ClientNo mapping found for 'com.ibm.jrom.JROMValue'
using encoding style 'http://schemas.xmlsoap.org/soap/encoding/';
```

```
nested exception is:
[SOAPException: faultCode=SOAP-ENV:Client; msg=No mapping found for
'com.ibm.jrom.JROMValue' using encoding style
'http://schemas.xmlsoap.org/soap/encoding/';
targetException=java.lang.IllegalArgumentException:
No mapping found for 'com.ibm.jrom.JROMValue' using encoding style
'http://schemas.xmlsoap.org/soap/encoding/'.]
[faultActorURI=/wsgwsoap1/soaprpcrouter]
...
```

- The WSDL is deployed to use **Deployed Classes**. In this case an empty message is returned. For example:

```
[Attributes={}] [faultCode=null] [faultString=null] [faultActorURI=null]
[DetailEntries=] [FaultEntries=]
```

Note: This is Apache SOAP behavior, and not something that the gateway can do anything about.

You are enabling operation-level authorization, but when you install wsgwauth.ear, an error message appears in the WebSphere Application Server administrative console detailing a JNDI problem.

Check that you entered, in the authorization session bean's 'EJB References', the correct JNDI name of the imported Web service enterprise bean. Note that this is case sensitive.

You are trying to have a Web services gateway send an SOAP/HTTPS message to another Web services gateway, and you are receiving a Malformed URLErrorException error.

The Web services gateway can invoke Web services that include https:// in their addresses, if the Java and WebSphere security properties have been configured to allow it. To check your security property settings, see the topic Invoking Web services over HTTPS

You deselect 'Authorization Policy - Control access to this service' from the deployment details for a Web service, and you find the service no longer works.

A number of tasks are required to disable security. Clearing the check box 'Authorization Policy - Control access to this service' will still leave WebSphere Application Server security in place, so basic authentication might still be required.

To disable security fully, use the WebSphere Application Server administrative console's Security Center to disable Global Security.

You experience problems with handling Document style SOAP messages.

You experience problems with handling SOAP messages with attachments.

You experience problems with a Web service that uses complex data types. The same service works fine, when not using the gateway.

You need to do one of two things to support Web services that use complex types in the Web services gateway:

- Set the message part representation for the service to **Generic classes**.
- Set the message part representation for the service to **Deployed Java classes**, and make the original classes available to the application server (either by updating the class path or by putting the JAR file somewhere like `WAS_HOME/lib/app`). If you do not have the original classes, you can use `wsdl2java` to generate java beans that can contain values of the complex type for the service, then compile the beans into a JAR file and make it available to the application server.

For more information on the factors to consider when choosing between these options, see Data type representation - choosing between Generic classes and Deployed Java classes.

Note:

- If your Web service has non-bean parameters (that is, it requires a custom serializer/deserializer) then it is not supported by the current release of the gateway.
- Not all complex types that are expressible in XML schema are supported by the current release of the gateway. For more information see Web services gateway - Supported types.
- See also Complex data types - mapping namespaces to packages.

Web services gateway messages

WebSphere system messages are logged from a variety of sources, including application server components and applications. Messages logged by application server components and associated IBM products start with a unique message identifier that indicates the component or application that issued the message. For more information about the message identifier format, see the topic Message Format.

The rest of this topic contains a list of the Web services gateway runtime system messages, with details of what each message means.

WSGW0001E: Channel name {0} from gateway configuration differs from that in JNDI: {1}

Explanation: The name specified for the channel does not match the name of the channel as defined within the EAR file.

User Response: Ensure that the channel name is specified correctly

WSGW0002E: Error storing endpoint address. Exception: {0}

Explanation: An unexpected exception occurred when storing the endpoint address for a channel.

User Response: Contact IBM Support

WSGW0003E: Error retrieving endpoint address. Exception: {0}

Explanation: An unexpected exception occurred when retrieving the endpoint address for a channel.

User Response: Contact IBM Support

WSGW0004E: Not used

Explanation:

User Response:

WSGW0005E: Error retrieving channel name. Exception: {0}

Explanation: An unexpected exception occurred when retrieving the channel name from JNDI.

User Response: Contact IBM Support

WSGW0006E: Error deploying service to {1}. Exception: {0}

Explanation: An unexpected error occurred deploying the service to the given component.

User Response: This error may be caused by a previous failure. Try

redeploying the service using a different gateway service name. If that fails, reinstalling the channel and gateway applications may remove the problem.

WSGW0007E: Error getting endpoint URL from channel {0}. Exception: {1}

Explanation: An unexpected error occurred generating the endpoint URL for the given channel.

User Response: Contact IBM Support

WSGW0008E: Could not determine default port name for target service {0}

Explanation: There are no ports in the WSDL defined for the target service that are supported by currently available WSIF providers or there is an error in the WSDL file associated with the port definition or a namespace it uses.

User Response: Either ensure that a WSIF provider is correctly configured for the port in the WSDL, or ensure that the WSDL contains correctly specified port information.

WSGW0009E: Failed to deploy service. Exception: {0}

Explanation: An unexpected error occurred trying to deploy the service.

User Response: Contact IBM Support

WSGW0010E: The namespaceURI attribute cannot be changed when there are active services

Explanation: The namespaceURI is used to generate WSDL for gateway services. If this global setting is changed then current WSDL becomes invalid.

User Response: Either remove all channels or all gateway services from the gateway configuration and retry the change.

WSGW0011E: Not used

Explanation:

User Response:

WSGW0012E: Not used

Explanation:

User Response:

WSGW0013E: Could not locate home {0}. Exception: {1}

Explanation: The specified home location could not be found in JNDI.

User Response: Ensure that the home location is specified correctly, and that it appears in JNDI.

WSGW0014E: Not used

Explanation:

User Response:

WSGW0015E: Could not create instance of class {0}. Exception: {1}

Explanation: The gateway failed to create an instance of the specified Java class.

User Response: Ensure that the Java class has a public constructor with no parameters.

WSGW0016E: Could not locate class {0}. Exception: {1}

Explanation: The gateway failed to locate the specified Java class.

User Response: Ensure that the Java class is visible to the gateway application's classloader.

WSGW0017E: Not used

Explanation:

User Response:

WSGW0018E: Not used

Explanation:

User Response:

WSGW0019E: Failed to clone definition. Exception: {1}

Explanation: An unexpected error occurred cloning a WSDL definition.

User Response: Contact IBM Support

WSGW0020E: Error while loading mapped type class {0}. Exception: {1}

Explanation: An error occurred while trying to load the given Java class which represents a type in the deployed WSDL for a target service.

User Response: Ensure that the Java class is visible to the gateway application's classloader.

WSGW0021E: Expected WSDL definition to contain a <wsdl:type> element with a schema from one of the '{0}', '{1}', or '{2}' namespaces

Explanation: Schema types in WSDL definitions must be declared using one of the specified XML Schema namespaces.

User Response: Update the WSDL definition to use the appropriate namespace.

WSGW0022E: Unexpected Schema->Java problem when parsing WSDL file.

Exception: {0}

Explanation: An unexpected exception occurred when parsing a WSDL file. This may be due to unsupported elements in the WSDL.

User Response: Contact IBM Support

WSGW0023E: Unexpected Schema->JROM problem when parsing WSDL file.

Exception: {0}

Explanation: An unexpected exception occurred when parsing a WSDL file. This may be due to unsupported elements in the WSDL.

User Response: Contact IBM Support

WSGW0024E: Channel {0} cannot be removed because it is being used by a deployed service

Explanation: Channels can only be removed when they are not in use by gateway services.

User Response: Remove the channel from gateway services to which it is deployed before removing the channel.

WSGW0025E: Target service identity cannot be specified as null

Explanation: A target service can only be selected using a non-null valid for the identity.

User Response: Modify the calling code to ensure that the target service identity value is never null.

WSGW0026E: Invalid gateway service name {0}. The name must be a valid XML Schema NCNAME.

Explanation: The name specified for the gateway service does not conform to the required definition.

User Response: Correct the gateway service name so that it is a valid XML Schema NCNAME.

WSGW0027E: Port {0} does not exist for target service {1}

Explanation: The requested port does not exist for the target service.

User Response: Ensure that a valid port is requested, or update the target service WSDL to contain a port of the requested name.

WSGW0028E: No binding for port {0} for target service {1}

Explanation: The requested port for the target service does not have a binding defined in the WSDL definition of the service.

User Response: Ensure that the target service WSDL has a binding for the requested port, or use a different port name.

WSGW0029E: No portType for binding {0} for port {1} for target service {2}

Explanation: The requested port for the target service does not have a portType defined in the WSDL definition of the service.

User Response: Ensure that the target service WSDL has a portType for the requested port, or use a different port name.

WSGW0030E: Not used

Explanation:

User Response:

WSGW0031E: Channel name {0} already exists

Explanation: The name specified for the channel is the same as that of a channel that is currently deployed.

User Response: Choose a different name for the channel, or remove the existing channel of the given name.

WSGW0032E: Channel name {0} not found

Explanation: No channel is currently deployed with the given name.

User Response: Use the name of a channel that is currently deployed.

WSGW0033E: Filter {0} cannot be removed because it is being used by a deployed service

Explanation: Filters can only be removed when they are not in use by gateway services.

User Response: Remove the filter from gateway services to which it is deployed before removing the filter.

WSGW0034W: Invocation of filter {0} failed. Exception: {1}

Explanation: An unexpected exception was thrown during processing of the given filter.

User Response: Contact IBM Support

WSGW0035E: Filter context version {0} not supported

Explanation: The context version that the filter requires is not supported by this version of the gateway.

User Response: Ensure that the filter is requesting the correct context version. It may be necessary to upgrade the gateway to support the filter.

WSGW0036E: Target service identity information {0} not matched for gateway service {1}

Explanation: A target service was requested by identity, but the identity information does not match any currently deployed target service.

User Response: Ensure that the identity information is correct, and that there is a target service deployed to the given gateway service with the right identity information.

WSGW0037E: Filter name {0} already exists

Explanation: The name specified for the filter is the same as that of a filter that is currently deployed.

User Response: Choose a different name for the filter, or remove the existing filter of the given name.

WSGW0038E: Filter name {0} not found

Explanation: No filter is currently deployed with the given name.

User Response: Use the name of a filter that is currently deployed.

WSGW0039E: Error loading state from {0}. Exception {1}

Explanation: An unexpected exception occurred loading the state of the gateway from the given location.

User Response: Ensure that the given location is visible to the gateway application.

WSGW0040E: Failed to convert definition to string. Exception: {0}

Explanation: An unexpected exception occurred converting a WSDL definition into a string in order to display it or return it to an application.

User Response: Contact IBM Support

WSGW0041E: Failed to save state. Exception {0}

Explanation: An unexpected exception occurred when saving the state of the gateway.

User Response: Contact IBM Support

WSGW0042W: No target services available to get service definition

Explanation: A request was made for the WSDL definition for the gateway service, however no target services have been defined for the gateway service, so it is not possible to generate a WSDL definition.

User Response: Deploy one or more target services to the gateway service.

WSGW0043E: Exception while executing operation {0} service {1}. Exception: {2}

Explanation: An unexpected exception occurred when passing a request on to a target web service.

User Response: Ensure that the gateway service and target service are correctly deployed (using the correct message part representation). Ensure that the target service is available and responds correctly to direct requests (i.e. not through the gateway).

WSGW0044E: Filter position {0} invalid

Explanation: The specified position for addition or removal of the filter was not valid.

User Response: Ensure a valid value is specified. The value should be -1, 0 or a positive integer.

WSGW0045E: Filter not found in list

Explanation: An attempt was made to remove a filter from a gateway service specifying -1 as the index, but the filter is not in the list at all.

User Response: Ensure that the correct filter is specified.

WSGW0046E: Channel {0} already defined for gateway service {1}

Explanation: The given channel has already been defined for the gateway service.

User Response: Ensure that the correct channel name is specified.

WSGW0047E: Channel {0} not defined for gateway service {1}

Explanation: The channel cannot be removed from the gateway service as it is not currently defined for the gateway service.

User Response: Ensure that the correct channel name is specified.

WSGW0048E: UDDI reference {0} already defined for gateway service {1}

Explanation: The given UDDI reference has already been defined for the gateway service.

User Response: Ensure that the correct UDDI reference name is specified.

WSGW0049E: UDDI reference {0} not defined for gateway service {1}

Explanation: The UDDI reference cannot be removed from the gateway service as it is not currently defined for the gateway service.

User Response: Ensure that the correct UDDI reference name is specified.

WSGW0050E: Target service with location {0} already defined for gateway service {1}

Explanation: The given target service location has already been defined for the gateway service.

User Response: Ensure that the correct target service location is specified.

WSGW0051E: Target service with location {0} not defined for gateway service {1}

Explanation: The target service location cannot be removed from the gateway service as it is not currently defined for the gateway service.

User Response: Ensure that the correct target service location is specified.

WSGW0052E: Target service with location {0} was not found for gateway service {1}

Explanation: The target service WSDL definition could not be obtained from the given location.

User Response: Ensure that the correct target service location is specified.

WSGW0053E: gateway service {0} cannot be removed as active entities and force not specified

Explanation: A gateway service with one or more target services, channels, filters or UDDI references cannot be removed.

User Response: Remove the target services, channels, filters and UDDI references from the gateway service.

WSGW0054E: An exported definition for gateway service {0} is not available as there are no defined channels for the service

Explanation: A request was made for the WSDL definition for the gateway service, however no channels have been defined for the gateway service, so it is not possible to generate a WSDL definition.

User Response: Deploy one or more channels to the gateway service.

WSGW0055E: Not used

Explanation:

User Response:

WSGW0056E: No default target service available for {0}

Explanation: The default target service location cannot be obtained for the gateway service as no target services are defined.

User Response: Ensure that one or more target services are defined for the gateway service.

WSGW0057E: No receiving channel name in context

Explanation: A request has reached the gateway that does not contain the receiving channel name in the context.

User Response: Contact the supplier of the channel application.

WSGW0058E: Channel {0} not defined for gateway service {1}

Explanation: A request has reached the gateway for the given service through a channel which is not defined for that service. The request is rejected.

User Response: If the channel should be valid for the service, add the channel, otherwise check that the client of the request is making a valid request. This exception may be thrown when a client is making a malicious attack.

WSGW0059E: gateway service {0} does not exist

Explanation: A request was made for a gateway service that does not exist.

User Response: Ensure that the correct gateway service name is specified.

WSGW0060E: gateway service {0} already exists

Explanation: An attempt was made to create a new gateway service using a name that is used by an existing gateway service.

User Response: Use a different name for the gateway service.

WSGW0061E: Could not find Service in UDDI registry {0} with parameters {1}, {2}, {3}

Explanation: The given parameters for UDDI lookup did not yield a match.

User Response: Ensure that the parameters are correct. Also ensure that the UDDI reference parameters are correct and correspond to those used to publish the service to UDDI.

WSGW0062E: Target service WSDL contains no <service> elements

Explanation: The target service WSDL could be loaded but does not contain a <service> element. This is necessary to be able to invoke the target service.

User Response: Ensure that the target service WSDL contains one or more <service> element.

WSGW0063E: Target service WSDL contains more than one service, and either target service name or namespace not specified

Explanation: When adding a target service to a gateway service, you must specify both the service name and namespace values if there is more than one <service> element in the target service WSDL.

User Response: Specify the target service name and namespace as well as the location.

WSGW0064E: Target service name {0} does not match service name in WSDL: {1}

Explanation: A target service name was specified that is not the same as any target service name in the WSDL at the given location.

User Response: Ensure that a valid target service name is specified.

WSGW0065E: Target service namespace {0} does not match service namespace in WSDL: {1}

Explanation: A target service namespace was specified that is not the same as any target service namespace in the WSDL at the given location.

User Response: Ensure that a valid target service namespace is specified.

WSGW0066E: Target service name {0} or namespace {1} not found in WSDL definition

Explanation: A target service name and namespace were both specified, but do not match any target service name and namespace combination in the WSDL at the given location.

User Response: Ensure that a valid target service name and namespace combination is specified.

WSGW0067E: UDDI reference {0} cannot be removed because it is being used by a deployed service

Explanation: UDDI references can only be removed when they are not in use by gateway services.

User Response: Remove the UDDI reference from gateway services to which it is deployed before removing the UDDI reference.

WSGW0068E: UDDI reference {0} already exists

Explanation: The name specified for the UDDI reference is the same as that of a UDDI reference that is currently deployed.

User Response: Choose a different name for the UDDI reference, or remove the existing UDDI reference of the given name.

WSGW0069E: UDDI reference {0} not found

Explanation: No UDDI reference is currently deployed with the given name.

User Response: Use the name of a UDDI reference that is currently deployed.

WSGW0070E: Invalid target service location type {0}

Explanation: The location type for the target service is not a valid value.

User Response: Ensure that a correct value is specified for the target service location type.

WSGW0071E: Failed to load URL definition from {0}

Explanation: The URL location specified was incorrect, or the WSDL it refers to cannot be loaded.

User Response: Ensure that the URL location is correct, and refers to a valid WSDL document.

WSGW0072E: Failed to load UDDI definition from {0}

Explanation: The UDDI location specified was incorrect, or the WSDL it refers to cannot be loaded.

User Response: Ensure that the UDDI location is correct, and refers to a valid WSDL document.

WSGW0073W: Not used

Explanation:

User Response:

WSGW0074E: Not used

Explanation:

User Response:

WSGW0075E: Failed to set gateway end point address. Exception {0}

Explanation: An unexpected exception occurred when automatically setting the gateway's end point address.

User Response: Contact IBM Support

WSGW0076E: Unable to access the gateway configuration bean. Exception {0}

Explanation: An unexpected exception occurred looking up the gateway's configuration bean in JNDI.

User Response: Restart the application server.

WSGW0077E: Failed to remove gateway configuration session. Exception {0}

Explanation: An unexpected exception occurred removing the session bean while access the gateway's configuration bean.

User Response: Contact IBM Support

WSGW0078E: Unable to access the gateway EndPoint bean. Exception {0}

Explanation: An unexpected exception occurred looking up the gateway's endpoint bean in JNDI.

User Response: Restart the application server.

WSGW0079E: Failed to remove endpoint session. Exception {0}

Explanation: An unexpected exception occurred removing the session bean while access the gateway's endpoint bean.

User Response: Contact IBM Support

WSGW0080E: Performance monitoring error. Exception {0}

Explanation: An unexpected exception occurred when recording performance monitoring information.

User Response: Contact IBM Support

WSGW0081E: Unexpected error in method {0}. Exception {1}

Explanation: An unexpected exception occurred in the given method.

User Response: Contact IBM Support

WSGW0082E: Unable to determine WAS security setting

Explanation: The WAS security setting could not be determined. It will be assumed that security is enabled.

User Response: No action required.

**WSGW0083W: Failed to authorize request for operation {0} on service {1}.
Exception {2}**

Explanation: Authorization of the given request failed. The request has been rejected.

User Response: Ensure that the required authorization bean has been generated for the given service, and that the correct authorization policy is defined.

WSGW0084W: Invocation of filter {0} version {1} failed. Exception {2}

Explanation: An exception was thrown during processing of the given filter. Processing of the request continues.

User Response: Investigate the reason for the exception being thrown by the filter. Refer to the documentation for the filter on how to resolve the problem.

WSGW0085E: Failed to publish service {0} to UDDI registry {1}. Exception: {2}

Explanation: An unexpected exception occurred when publishing the given service to a UDDI registry.

User Response: Ensure that the properties of the gateway service and UDDI reference are specified correctly.

WSGW0086E: Failed to unpublish service {0} from UDDI registry {1}. Exception: {2}

Explanation: An unexpected exception occurred when unpublishing the given service from a UDDI registry.

User Response: Ensure that the properties of the gateway service and UDDI reference are specified correctly.

WSGW0087I: Published service {0} to UDDI registry {1}

Explanation: The service was successfully published to the UDDI registry.

User Response: None

WSGW0088I: Unpublished service {0} from UDDI registry {1}

Explanation: The service was successfully unpublished from the UDDI registry.

User Response: None

WSGW0089I: No MessageWarehouse registered. Requests will not be logged

Explanation: A MessageWarehouse implementation was not found at the expected location in JNDI, so none is being used.

User Response: If a MessageWarehouse has been implemented, ensure that it is bound to JNDI at the correct location.

WSGW0090I: No ExceptionHandler registered. Exceptions will not be handled

Explanation: An ExceptionHandler implementation was not found at the expected location in JNDI, so none is being used.

User Response: If an ExceptionHandler has been implemented, ensure that it is bound to JNDI at the correct location.

**WSGW0091I: Usage: java -jar GenAuth -DWAS_HOME=<was.install.directory>
<HostName> <ServiceName>**

where <was.install.directory> is the location of the WebSphere installation directory and <HostName> is the url pointed to the installation of the gateway and <ServiceName> is the name of the deployed gateway service. (Please note the ServiceName is case sensitive).

For example

```
java -jar GenAuth.jar -DWAS_HOME=  
c:\websphere\AppServer http://host.machine.name.com/wsgw ServiceName
```

Successful execution will generate a file named <ServiceName>.ear

Explanation: Usage statement. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0092I: Retrieving Service :

Explanation: Progress message indicating that the service definition is being retrieved. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0093I: Retrieving Port Type :

Explanation: Progress message indicating that the port type information is being retrieved. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0094I: Retrieving Methods :

Explanation: Progress message indicating that method information is being retrieved. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0095I: Making Directory :

Explanation: Progress message indicating that a directory is being created. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0096I: Using Directory :

Explanation: Progress message indicating that a directory is being used. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0097I: About to compile....

Explanation: Progress message indicating that a compilation is about to start. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0098I: Command Status :

Explanation: General command status message. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0099I: About to create jar....

Explanation: Progress message indicating that a JAR file is about to be created. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0100I: About to create ear....

Explanation: Progress message indicating that an EAR file is about to be created. This message is used by the WSGWAuthGen command line utility.

User Response: No action required.

WSGW0101E: Error retrieving port from service {1}

Explanation: An error occurred retrieving the port from the service in the WSDL. This message is used by the WSGWAuthGen command line utility.

User Response: Ensure that the service name is specified correctly and is deployed to the gateway with at least one target service and one channel.

WSGW0102E: Error retrieving service {0}

Explanation: An error occurred retrieving the service. This message is used by the WSGWAuthGen command line utility.

User Response: Ensure that the service name is specified correctly and is deployed to the gateway with at least one target service and one channel.

WSGW0103E: Exception while retrieving service definition from URL: {0}/ServiceDefinition?name={1}. Exception: {2}

Explanation: An unexpected exception occurred retrieving WSDL from the given location. This message is used by the WSGWAuthGen command line utility.

User Response: Ensure that the service name is specified correctly and is deployed to the gateway with at least one target service and one channel.

WSGW0104E: Error retrieving methods from service {0}

Explanation: An unexpected exception occurred retrieving the methods that correspond to operations on the service.

User Response: Contact IBM Support

WSGW0105E: Error retrieving WAS_HOME environment variable

Explanation: The value of the WAS_HOME environment variable could not be retrieved.

User Response: Ensure that the WAS_HOME variable is set correctly in the environment under which the command is being executed.

WSGW0106E: Error compiling files

Explanation: An unexpected error occurred compiling the generated Java files.

User Response: Contact IBM Support

WSGW0107E: Error executing JAR command

Explanation: An unexpected error occurred generating a JAR file.

User Response: Contact IBM Support

WSGW0110E: A client attempted to load imported URL {0} for gateway service {1}. This URL is not imported by the definition for that service.

Explanation: An attempt was made to use the gateway's import mapping servlet to load information from a URL that does not correspond to one that is referenced by the WSDL definition for that service.

User Response: Ensure that the client is making a valid request. This may be a malicious attempt to obtain information that the client does not have access to.

WSGW0111W: Unsupported elements within the WSDL definition for target service {0} were ignored. The functionality of this service may be compromised.

Explanation: In order to be able to use the given WSDL definition within the gateway, certain elements of the definition were ignored.

User Response: Refer to the service provider's documentation to determine whether this will affect the use of the service.

WSGW0120E: Exception while removing ConversationPart {0} from Correlation Service. Exception {1}

Explanation: An unexpected exception occurred when using the Correlation Service.

User Response: Contact IBM Support

WSGW0121E: Exception while accessing ConversationPart {0} from Correlation Service. Exception {1}

Explanation: An unexpected exception occurred when using the Correlation Service.

User Response: Contact IBM Support

WSGW0122E: Exception while storing Serializable {0} at Correlation Service. Exception {1}

Explanation: An unexpected exception occurred when using the Correlation Service.

User Response: Contact IBM Support

WSGW0123E: Exception while storing Serializable {0} with id {1} at Correlation Service. Exception {1}






Explanation: An unexpected exception occurred when using the Correlation Service.

User Response: Contact IBM Support

Web services gateway: Resources for learning

Use the following links to find supplementary information about getting started with the Web services gateway. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

-  **WebSphere Version 5 Web services Handbook** (<http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246891.html?Open>) . This Redbook illustrates with suitable demonstration applications how Web services can be implemented using the IBM product portfolio, especially WebSphere Application Server Version 5 and WebSphere Studio Application Developer Version 5. It includes chapters on the gateway and on Web service security.
-  **Samples Central** (<http://www.ibm.com/websphere/developer/library/samples/AppServer.html>) .The gateway samples, and documentation on how to use them, are available through the Samples Central page of the IBM WebSphere Developer Domain Web site.
-  **InfoCenter for WebSphere Application Server Edge components** (<http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html>) . This InfoCenter contains a library of PDF online books covering all aspects of the WebSphere Application Server Edge components. gateway clustering builds upon the load balancing capabilities of these components.
-  **The Web Services Security (WS-Security) specification** (<http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>) . A major area of gateway security is based upon this emerging standard.
-  **The IBM Web services gateway: Technical Overview** (<http://www-3.ibm.com/software/integration/busconn/gateway.html>) . A different version of the gateway is available as a component of a product called IBM WebSphere Business Connection. This brief technical summary from WebSphere Business Connection applies equally well to the version of the gateway in WebSphere Application Server.

For supplementary information about Web services in general, see Web services: Resources for learning.

The gateway builds on the Web Services Invocation Framework (WSIF), which allows the gateway to pass on Web service invocations to any WSDL-defined Web service. For supplementary information about WSIF, see [WSIF: Resources for learning](#).