IBM WebSphere Application Server Network Deployment, Version 5

# Servers

**Note**

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

**Compilation date: November 21, 2002**

# Contents

## Trademarks and service marks

# 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.

You can (configure one or more application servers) and enhance the operation of an application server using:
- (transports)
- (custom services)
- (command-line information) that passes to a server when it starts or initializes
- Settings that (improve the use of the Java virtual machine (JVM))

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

**Clusters**

Clusters are groupings of servers. Each server in a cluster is a *member* of the cluster. Using clusters simplifies administration in that applications installed to a cluster are automatically installed to each member in the cluster. Likewise, applications removed from a cluster are automatically removed from each member in the cluster. When you (create a cluster with multiple members), each member contains the same applications and, thus, can service the same client requests.

Using clusters can optimize the distribution of client processing tasks (load balancing) and improve application availability (failover).

Clusters can help balance loads in that clustering enables multiple servers to service the same client request; a request from a given client can be routed to any of the cluster members. Thus, rather than having all client requests handled by a single application server, client work can now be distributed across all members of a cluster. This enables systems to be scaled up to serve a higher client load than could be provided by a single server. Further, you can configure multiple cluster members on the same physical machine (vertical scaling), configure multiple cluster members on different physical machines (horizontal scaling), or do both. Finally, you can specify the amount of work targeted to each member of a cluster. For example, you can distribute client tasks according to the capacities of the different machines in the enterprise.

That multiple servers can service the same client request is also the basis for failover support. If a server fails while processing a client request, the failed request can be rerouted to any of the remaining cluster members. In fact, several servers could fail, and as long as at least one cluster member is running, client requests can continue to be serviced.

**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. For Network Deployment and Enterprise Extensions, the JMS functions (of JMS providers) within the administration domain are served by one or more JMS servers. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain.

**Web services servers**

The Web services components included with this product version build upon the Apache Simple Object Access Protocol (SOAP) 2.3-based capabilities delivered with Version 4.0.x.

New in Network Deployment are the following features:
- A private Universal Description, Discovery and Integration (UDDI) Registry, implementing Version 2.0 of the UDDI specification
- A Web Services Gateway for providing gateway access to existing Web services

# 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 process execution statements for starting or initializing a UNIX process, monitoring policies to track the performance of a process, process logs to which standard out and standard error streams write, 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. On the Network Deployment product, you can also create a new application server when you (add a cluster member to a server cluster). 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 it 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 single-server (base) 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, use the wasadmin tool, or use command line tools such as startServer and stopServer.

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. The Network Deployment product also shows the status of the application servers. The status indicates whether a server is started, stopped, or unavailable.

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

   To view product information for a server:

   a. Ensure that the server is running.
   b. Go to 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 server, the version and build levels for the products, the build dates, and any e-fixes applied to the server.
3. Create an application server. Click **New** and follow the instructions on the Create New Application Server page.
4. Start your application server.
5. Monitor the running of application servers.
6. Stop your application server.
7. **(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 the server you want deleted.
   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.

The Network Deployment product also shows the status of the application servers. The status indicates whether a server is running, stopped, or encountering problems.

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 |

**Initial State:** Specifies the desired state of the application server when its containing process (node) starts. The options are *Started* and *Stopped*. The default is *Started*.

| | |
|---|---|
| Data type | String |
| Default | Started |

**Application Classloader Policy:** Specifies whether to use a single classloader to load all applications or to use a different classloader for each application.

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

| | |
|---|---|
| Data type | String |
| Default | MULTIPLE |

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

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

| | |
|---|---|
| Data type | String |
| Default | PARENT_FIRST |

**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 | *host_name*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.

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

**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 |

**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.

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

**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.

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

**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. 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.

| | |
|---|---|
| Data type | Integer |
| Default | 50 |

**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 current server configuration's process definition settings. The node agent for the node on which the application server resides must be running before you can start the application server.

Note that after you start a server, other processes might not discover the running server immediately. Application servers are discovered by the node agent and node agents are discovered by the deployment manager. Node agents usually can discover local application servers quickly but it can take a deployment manager from 2 to 60 seconds to discover node agents.

**Starting a server from a command line**

To start a server, run the startServer command.

If the node agent for the node on which the application server resides is not running, run the startNode command and then run the startServer command.

**Starting an application server using the administrative console**

1. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
2. If the node agent for the node on which the application server resides is not running, click **Restart** or **Restart all Servers on Node** on the Node Agents page to start the node agent.
3. Place a checkmark in the check box beside the application server that you want started and click **Start**.
4. View the **Status** value and any messages or logs to see whether the server starts.

**Starting a server for tracing and debugging**

To start the server with standard Java debugging enabled:

1. 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**.
2. 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**.
3. Save the changes to a configuration file.
4. Stop the server.
5. Start the server again (see above).

## Running servers as non-root using the console

By default, WebSphere Application Server servers use a root ID. A server can be run using a non-root ID if security file system permissions grant all users of a certain group writable access to main WebSphere Application Server directories and the user ID and group ID for the server ″run as″ the user and group.

Steps for this task

1. Specify user and group ID values for the **Run As User** and **Run As Group** settings for a server
   a. Go to the Process Execution page for the server you want to run as non-root. Click **Servers > Application Servers >** *server_name* **> Process Definition > Process Execution**.
   b. For **Run As User**, specify a user name for the process to run as.
   c. For **Run As Group**, specify a group name for the process to run as.
2. Using operating system tools, create a set of users that are all in the group.
3. Using operating system tools, change the permissions of the WebSphere Application Server installation root (*install_root*) directory.
   a. Change the group owner to the group.
   b. Make the following files under the *install_root* directory writable by the group:
      - Log files
      - All files and subdirectories below the tranlog directory
      - All files and subdirectories below the config/temp directory
4. From the user ID, run the startServer command to start the server.

### 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.

   One way is using the Logging and Tracing page of the administrative console. Click **Troubleshooting > Logs and Trace** in the console navigation tree to access the page.

   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. If messages indicate that a server has stopped running, use the Application Servers page to try restarting the server. If a cluster of servers has stopped running, use the (Server Cluster page) to try (restarting the cluster). If the status of an application server is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.
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 a server stops a server process based on the current server configuration's process definition settings.

**Stopping a server from a command line**

To stop a server, run the stopServer command.

**Stopping an application server using the administrative console**

1. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
2. Place a checkmark in the check box beside the application server that you want stopped and click **Stop**.
3. Confirm that you want to stop the server.
4. View the **Status** value and any messages or logs to see whether the server stops.

   A warning message displays if you are stopping the server that is running the administrative application.

## 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**

For applications in a test or development environment (in other words, a non-production environment), you can use the internal HTTP transport system to serve servlets without an Web server plug-in. Simply use the internal HTTP transport port (typically on port 9080).

For example, to serve a servlet (*servlet_path_name*) without an HTTP server, use the URL: `http://`*`server_name`*`:`*`port`*`/`*`servlet_path_name`*

with *port* being the internal transport port number (typically `9080`) and *server_name* being `localhost` if the application server is on the local machine.

For a production environment, do not use the internal transport, as it lacks the performance available when using a Web server plug-in.

At times, you might be able to configure the internal transport to use a port other than 9080. 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 your 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.

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. 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.

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

### 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: Manually editing transport settings in the server.xml file

WebSphere Application Server Version 4.x has several transport properties that are not shown in the settings page for an HTTP transport:

**ConnectionIOTimeout**
Specifies the maximum number of seconds to wait when trying to read or write data during a request.

**ConnectionKeepAliveTimeout**
Specifies the maximum number of seconds to wait for the next request on a keep alive connection.

**MaxKeepAliveConnections**
Specifies the maximum number of concurrent keep alive (persistent) connections across all HTTP transports. The default value is 90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep alive connections so that there are threads available to handle new incoming connect requests.

**MaxKeepAliveRequests**
Specifies the maximum number of requests which can be processed on a single keep alive connection.

To specify values for these transport properties, you can edit the <*properties*> settings in the server.xml file shown in **bold** font below. After editing the server.xml file, restart the server and regenerate the Web server plug-in. As to each of the new properties, whatever value you specify for a first transport is applied globally to all other HTTP transports in the cell.

```
<components xmi:type="applicationserver.webcontainer:WebContainer"
               xmi:id="WebContainer_1" enableServletCaching="false">
 <stateManagement xmi:id="StateManageable_5" initialState="START"/>
  <properties xmi:id="WebContainer_Property_1" name="MaxConnectBacklog" value="50"/>
  <properties xmi:id="WebContainer_Property_2"
       name="MaxKeepAliveConnections" value="45"/>
  <properties xmi:id="WebContainer_Property_3"
               name="MaxKeepAliveRequests" value="100"/>
  <properties xmi:id="WebContainer_Property_4"
               name="ConnectionIOTimeout" value="5"/>
  <properties xmi:id="WebContainer_Property_5"
               name="ConnectionKeepAliveTimeout" value="5"/>
 <services xmi:type="applicationserver.webcontainer:SessionManager"
          xmi:id="SessionManager_1" enable="true" enableUrlRewriting="false"
          enableCookies="true" enableSSLTracking="false"
          enableProtocolSwitchRewriting="false" enableSecurityIntegration="false"
          sessionPersistenceMode="NONE" allowSerializedSessionAccess="false"
          accessSessionOnTimeout="true" maxWaitTime="5">
    <defaultCookieSettings xmi:id="Cookie_1" name="JSESSIONID" domain=""
          maximumAge="-1" path="/" secure="false"/>
    <sessionDatabasePersistence  xmi:id="SessionDatabasePersistence_1"
            datasourceJNDIName="jdbc/Sessions"
            userId="db2admin" password="db2admin" db2RowSize="ROW_SIZE_4KB"
            tableSpaceName=""/>
    <tuningParams xmi:id="TuningParams_1" usingMultiRowSchema="false"
            maxInMemorySessionCount="1000" allowOverflow="true"
            invalidationTimeout="30" writeContents="ONLY_UPDATED_ATTRIBUTES"
            writeFrequency="TIME_BASED_WRITE" writeInterval="120"
            scheduleInvalidation="false">
      <invalidationSchedule xmi:id="InvalidationSchedule_1" firstHour="14"
              secondHour="2"/>
    </tuningParams>
 </services>
  <threadPool xmi:id="ThreadPool_2" minimumSize="10" maximumSize="50"
              inactivityTimeout="3500" isGrowable="false"/>
  <transports xmi:type="applicationserver.webcontainer:HTTPTransport"
              xmi:id="HTTPTransport_1" sslEnabled="false">
    <address xmi:id="EndPoint_1" host="" port="9080"/>
  </transports>
  <transports xmi:type="applicationserver.webcontainer:HTTPTransport"
              xmi:id="HTTPTransport_2" sslEnabled="true"
              sslConfig="DefaultSSLSettings">
    <address xmi:id="EndPoint_2" host="" port="9443"/>
  </transports>
  <transports xmi:type="applicationserver.webcontainer:HTTPTransport"
              xmi:id="HTTPTransport_3" sslEnabled="false">
    <address xmi:id="EndPoint_3" host="" port="9090"/>
  </transports>
  <transports xmi:type="applicationserver.webcontainer:HTTPTransport"
              xmi:id="HTTPTransport_4" sslEnabled="true"
              sslConfig="DefaultSSLSettings">
    <address xmi:id="EndPoint_4" host="" port="9043"/>
  </transports>
</components>
```

## 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 ConfigServer described in the Javadoc file at ../javadoc/ae/com/ibm/websphere/management/configservice /ConfigService.html.

   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 defintions 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 defintion 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 defintion, 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 process execution statements for starting or initializing a UNIX process.
4. **(Optional)** Specify monitoring policies to track the performance of a process.
5. **(Optional)** Specify process logs to which standard out and standard error streams write. Complete this step if you do not want to use the default file names.
6. **(Optional)** Specify name-value pairs for properties needed by the process definition.
7. Stop the application server and then restart the server.

8. 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 starting or initializing a process.

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

### 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 admininistor 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 are directed. The file name can include a symbolic path name defined in the variable entries.

To direct server output to the administrative console or to the process that launched the server, either delete the value for this property or specify `console`.

| | |
|---|---|
| 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.

| | |
|---|---|
| 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. The default is 64 for OS/400 and zero (0) for all other platforms.

| | |
|---|---|
| Data type | Integer |
| Default | 64 for OS/400, 0 for all other platforms |

## Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes. The default is zero (0) for OS/400 and 256 for all other platforms.

| | |
|---|---|
| Data type | Integer |
| Default | 0 for OS/400, 256 for all other platforms |

## 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.

| | |
|---|---|
| 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. The default is not to disable JIT support.

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

### 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.

## 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 model and decisions
- Programming instructions and examples
- Programming specifications
- Administration

**Programming model and decisions**

- **Exploiting the Java Virtual Machine** (http://www.develop.com/downloads/DevWPJav.pdf)

**Programming instructions and examples**

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

**Programming specifications**

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

**Administration**

- **IBM WebSphere Administration** (http://www.mcgraw-hill.co.uk/html/0072223154.html)
- **Listing of all IBM WebSphere Application Server Redbooks** (http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere)
- **IBM WebSphere V4.0 Advanced Edition Handbook** (http://www.redbooks.ibm.com/abstracts/sg246176.html)
- **WebSphere 4.0 Installation and Configuration on the IBM iSeries Server** (http://publib-b.boulder.ibm.com/Redbooks.nsf/9445fa5b416f6e32852569ae006bb65f/7b1a07251256f08b85256b750067aee1?OpenDocument)
- **Redbook on Backing up WebSphere Application Server with Tivoli Storage Management** (http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/redp0149.html?Open)

# 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. **(Optional)** Adjust (thread-pool settings) used by the ORB for handling IIOP connections.
3. 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.

## Object Request Broker tuning guidelines

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 Object Request Broker (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 will wait before timing out, setting an ORB time-out requires some 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 threshhold 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

You can also 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.

## 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**.

## 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.

| | |
|---|---|
| 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.IIOPTunnelServlet` or, for applets, `http://applethost:port/servlet/com.ibm.CORBA.services.IIOPTunnelServlet`. 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

When enabled, this specifies that the ORB is to pass parameters by reference instead of by value, which bypasses a copy operation. Enable this property with caution, because unexpected behavior might occur.

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`) |

## 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 that administrative console page, click **Servers > Application Servers >** *serverName* **> ORB Service > Custom Properties**.

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

The page might already include Secure Socket 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. The default value is 2809. 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.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. The default value is 1024 bytes. The valid range is from 64 to the largest value of the 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. The default value is the next available system-assigned port number. The valid range is 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**
The property changes the maximum queue length for server incoming TCP/IP connection requests. If a connection requests arrives when the queue is full, the connection is refused. The valid range is between 50 and the maximum Java int value. The default value is 50.

**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.

If needed for locale support, you can also set and monitor properties for codeset conversion. For details, see ″Character codeset conversion support for the Java Object Request Broker service.″

## 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:O=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/IIOP 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/IIOP 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. By default, the ORB uses `UCS2`. 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 | |
| ISO8859_15_FDIS | 0x0001000F | |

| Java name | OSF registry ID | Comments |
|---|---|---|
| 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)

  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)

# 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 for which you will define a cluster member.
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.)
10. **(Optional)** Tune the behavior of the workload management run time. If your application is experiencing problems with timeouts or your network experiences extreme latency, change the timeout interval for the com.ibm.CORBA.RequestTimeout property. Or, if the workload management state of the client is refreshing too soon or too late, change the interval for the com.ibm.websphere.wlm.unusable.interval property.

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 reply 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.
- The WebSphere Application Server Session Manager, which is part of each application server, stores client session information and takes session affinity and server affinity into account when directing client requests to the cluster members of an application server. The workload management service takes server affinity and transaction affinity into account when directing client requests among the cluster members of an application server.

## 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 route requests routed to the local node if possible, place a checkmark in the **Prefer local enabled** check box.
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 include an existing server in the cluster. You can create an empty cluster and then add server members to it. To create an empty cluster, do not include an existing server in this cluster. Or, you can create a cluster based on an existing server member, then create and add additional server members to the cluster. To create a cluster based on an existing server, choose **Select an existing server to add to this cluster** and then select the server from the drop-down list.
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 100.
   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*.

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

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

**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 |

**wlcID:** 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**.

### 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.

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 |

**Available Servers:** 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.

## 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=`***`timeout_interval`*
   : 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=`***`interval`*
   : 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:
- Planning, business scenarios, and IT architecture
- Programming model and decisions
- Programming instructions and examples

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

- **WebSphere Application Server V5.0: Architecture and Overview** (http://developerworks.cybercentral.com/ibm0502/amt /ibmpresentations/683_1.pdf)

**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** (ftp://vadd1:ua33pcww@207.25.253.53/1/wsdd/pdf/modjeski.pdf)

**Programming instructions and examples**

- **WebSphere Application Server education** (http://www.ibm.com/software/webservers/learn/)
- **IBM WebSphere Administration** (http://www.mcgraw-hill.co.uk/html/0072223154.html)
- **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)
- **WebSphere Application Server Bible** (http://www.wiley.com/cda/product/0,,0764548964%7Ctoc%7C2948,00.html)

# Chapter 5. IBM WebSphere UDDI Registry

Welcome to the IBM WebSphere UDDI Registry.

Use the table of contents on the left to view the various topics for a specific product or technology. Select the topic you are interested in to either open documentation locally or find information about how to locate documentation.

- ″UDDI Registry Terminology″
- ″UDDI Registry Definitions″
- ″An overview of IBM UDDI Registries″
- ″Migrating from the IBM WebSphere UDDI Registry on WebSphere Application Server 4.0″
- ″Installing and Setting up a UDDI Registry″
- ″Reinstalling the UDDI Registry application″
- ″Removing the UDDI Registry application from a deployment manager cell″
- ″Removing the UDDI Registry application from a single appserver″
- ″Configuring the UDDI Registry″
- ″Administering the UDDI Registry″
- ″UDDI user console″
- ″SOAP Application Programming Interface for the UDDI Registry″
- ″UDDI Registry application programming interface″
- ″UDDI EJB Interface for the UDDI Registry″
- ″UDDI Troubleshooting Tips″
- ″Messages″
- ″Running the UDDI Samples″
- ″Installation Verification Program (IVP)″
- ″Reporting Problems with the IBM WebSphere UDDI Registry″
- ″Feedback″

## UDDI Registry Terminology

**Syntax**

When reference is made to the directory location of the WebSphere Application Server it is referred to as **<WebSphere-install-dir>** and , the directory location of the WebSphere Deployment manager as **<DeploymentManager-install-dir>**. These translate, by default, to the following locations:

**Windows**

**<WebSphere-install-dir>**
C:\Progra~1\WebSphere\AppServer\

**<DeploymentManager-install-dir>**
C:\Progra~1\WebSphere\DeploymentManager\

**Unix Platforms**

**(Linux/Solaris) - <WebSphere-install-dir>**
: /opt/WebSphere/AppServer/

**<(AIX) - <WebSphere-install-dir>**
: /usr/WebSphere/AppServer/

**<DeploymentManager-install-dir>**
: /opt/WebSphere/DeploymentManager/

### UDDI Registry Definitions

**Syntax**

**bindingTemplate**
: Technical information about a service entry point and construction specifications.

**businessEntity**
: Information about the party who publishes information about a family of services.

**businessService**
: Descriptive information about a particular service.

**publisherAssertion**
: Information about a relationship between two parties, asserted by one or both.

**tModel**
: Descriptions of specifications for services or taxonomies. The basis for technical fingerprints. This is short for technical model

## An overview of IBM UDDI Registries

The Universal Description, Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web Services. The term 'Web service' describes specific business functionality exposed by a company, usually through an Internet connection, to allow another company, or its subsidiaries, or software program to use the service.

**Universal Business Registries (IBM UBR)**

The IBM Universal Business Registry is one of a group of web-based registries that expose information about a business or other entity and its technical interfaces (or APIs). These registries are run by multiple Operator Sites, and can be used by anyone who wants to make information available about one or more businesses or entities, as well as anyone that wants to find that information. Although there are Universal Business Registries (sometimes referred to as 'public UDDI registries') hosted worldwide, including one hosted by IBM, enterprises may wish to host their own internal registries behind their firewall to better manage their internal implementation of Web Services.

For more detailed information about UDDI in general visit http://www.uddi.org

**IBM WebSphere UDDI Registry**

The IBM WebSphere UDDI Registry is a directory for Web Services that is implemented using the UDDI specifications. In contrast with the IBM UBR, this component of WebSphere Network Deployment is a product offering for companies or industries to implement.

A critical component of IBM's dynamic e-business infrastructure, IBM WebSphere UDDI Registry solves the problem of discovery of technical components for an enterprise and its partners by:

- Providing control, flexibility and confidentiality so that an enterprise can protect its e-business investments
- Increasing efficiency by making it easier to identify technical assets
- Leveraging existing infrastructures

For example, the IBM WebSphere UDDI Registry could be used in the following way within a large enterprise:

A company has a legacy application that provides phone numbers and Human Resources (HR) information of employees. This is turned into a Web Service and published to the registry. A developer in the same company needs to write an application for a procurement function that also needs to provide HR information to the supplier. The application should allow the supplier to have access to the employee account codes once the employee provides his name or serial number. Before Web Services, the developer had three choices:

1. Would not have known about the similar application
2. Knew about it but could not reuse due to technical barriers
3. Knew about it and reused only after significant time and negotiation

With UDDI, the developer can search for the ″web service″ and reuse the existing technical component in his new application for the supplier in a matter of minutes. The developer saves time and gets the application up and running sooner than he would have otherwise — increasing efficiency and saving the company time and money. The IBM WebSphere UDDI Registry is the first version 2 standard-compliant UDDI registry for private enterprise work. The IBM WebSphere UDDI Registry:

- Supports the public UDDI V2.0 standard
- Leverages the proven, reliable WebSphere Application Server technology
- Uses a relational database, such as DB2, for its persistent store.

## Migrating from the IBM WebSphere UDDI Registry on WebSphere Application Server 4.0

Before you begin

If you have previously installed the IBM WebSphere UDDI Registry V1.1 (or later refreshes) on WebSphere Application Server V4.0, then you should take the following steps in order to migrate to the UDDI Registry that is available as part of WebSphere Application Server for Network Deployment V5.0.

1. If you have made any changes to the configuration properties in the file uddi.properties, which is located in the properties subdirectory of your WebSphere AppServer install tree, then you should make a copy of this file (or make a note of all the changes), so that you will be able to reapply the changes to the file after you have installed WebSphere Application Server V5.0.
2. If you have made any other configuration changes, for example to the GUI style sheets or to the SOAP interface properties, you should make a note of them, and re-apply them after upgrading to the new UDDI version.

3. Uninstall the IBM WebSphere UDDI Registry using Add/Remove Programs on Windows platforms, or rpm -e IBMWebSphere-UDDI on Unix Platforms. This will remove the application, but will preserve the UDDI Registry database.
4. Please note that, if you wish to continue using DB2 as the persistence store for the UDDI Registry, and if you have any data in the UDDI Registry which you wish to preserve, then you should not run the DB2 setup wizard to create the DB2 version of the UDDI Registry database, but will instead be able to continue using the database that you already have. However, if you do run the wizard, you will be prompted whether you wish to keep the database or overwrite it.
5. You can now follow the instructions on installing (or upgrading) IBM WebSphere Application Server V5 and ″Installing the UDDI Registry component″.
6. After completing the install procedure, you should edit the uddi.properties file to reflect any changes that you require to the configuration properties. You should **not** replace this file with your previous copy from the IBM WebSphere UDDI Registry V1.1 or later refreshes. You can also re-apply any other configuration changes as necessary.
7. If you are migrating from version 1.1 of the IBM WebSphere UDDI registry, then there have been a few minor changes to the EJB interface which means that you may need to modify your EJB client applications. If you are migrating from later refreshes of the IBM WebSphere UDDI Registry (such as version 1.1.1), then you should not need to make any changes to your EJB clients.

   The changes for version 1.1 are when saving a new service through the EJB interface either using **saveBusiness** or **saveService**, you should not set the **serviceKey** before calling **saveBusiness** or **saveService**.

In the IBM WebSphere UDDI Registry V1.1 the term ′service type′ was used to refer to a ′technical model′ or ′tModel′, for example, in various of the panels in the User Console. This term has now been replaced by the term ′technical model′.

## Installing and Setting up a UDDI Registry

Before you begin

If you wish to use the UDDI User Console using Internet Explorer as your web browser, and using SSL, then you must use Internet Explorer V5.5 with SP2 and security fix Q321232 (which must be applied in that order).

If you are migrating from the IBM WebSphere UDDI Registry product that was available to run on WebSphere Application Server 4.0, then you need to read the section Migrating from the IBM WebSphere UDDI Registry on WebSphere Application Server 4.0 before installing the new product.

**Choice of database product to be used as the persistence store**

The UDDI Registry application can use either DB2 or Cloudscape as the persistence store for the registry data.

- If you plan to use the UDDI Registry in production then you must use DB2 as your persistence store.
- If you plan to use the UDDI Registry for development and testing purposes, then you can choose to use Cloudscape as your persistence store. Please note that Cloudscape is not intended for production purposes.

As part of the installation of the IBM WebSphere Application Server with Network Deployment option, you are given the option to install the ″UDDI Registry″, which is shown under Web Services. Having selected the UDDI Registry and installed the various files that make up the application, you have two choices as to the environment in which you run it:

1. Install the UDDI Registry application into the deployment manager cell using DB2 or Cloudscape as the database in which the UDDI Registry data will be held, selecting one of the application servers within the cell in which to run the UDDI Registry.
2. Install the UDDI Registry application directly into an application server using DB2 or Cloudscape as the database in which the UDDI Registry data will be held. Please note that, if you choose this option, then the application server on which you run the UDDI Registry must not be added into a deployment manager cell, as this would cause the file synchronization within the cell to remove the application.

In most cases you will probably choose option 1, and install the UDDI Registry into a deployment manager cell, but you might find that option 2, to install the UDDI Registry into a standalone application server, is useful for development or test purposes.

**Note:**

1. Several WebSphere commands are used during the following procedures, some of which must execute on the DeploymentManager and some of which must execute on the target Application server. The instructions below will distinguish which is appropriate for each command. The WebSphere commands will be found in the bin subdirectory of the appropriate WebSphere install tree. In order to ensure correct operation of these commands, you will need to do one of the following:
   - Ensure that the appropriate bin subdirectory is in your path prior to executing the command
   - Change directory to the appropriate bin subdirectory
   - Fully qualify the path to the commands
2. It is recommended that you use the version of java shipped with WebSphere found in the java/bin subdirectory.

The following table lists the UDDI Registry files, and the locations into which they are placed by the install. The ″Location″ column shows the subdirectory under the WebSphere DeploymentManager install directory. For example, if you had installed IBM WebSphere Application Server with Network Deployment option onto a machine running Windows, and had used the default directory, then a location of ″installableApps″ would mean that the file had been placed into the `C:\Progra~1\WebSphere\DeploymentManager\installableApps` directory. For Windows platforms, read the ″/″ directory separator in the location column as a ″\″ directory separation character.

| Files | Purpose | Location |
|---|---|---|
| uddi.ear | The UDDI Registry application itself, which is packaged and runs as an Enterprise Application | installableApps |
| uddi.properties | Provides configuration properties for the UDDI Registry application | properties |

| | | |
|---|---|---|
| uddiresourcebundles.jar | Contains system messages for the UDDI Registry application | lib |
| uddicloudscapeuserfunc.jar | Contains functions that are used by Cloudscape if the Cloudscape database is used with the UDDI Registry | lib |
| setupuddi.jacl | Admin script to create a JDBC driver and datasource for the UDDI Registry, and to install the UDDI Registry application in a DeploymentManager Cell | UDDIReg/scripts |
| setupuddimessages.jar | Contains setup and install messages for the UDDI Registry application | lib |
| removeuddi.jacl | Admin script to undo the effects of setupuddi.jacl | UDDIReg/scripts |
| appserverremoveuddi.jacl | Admin script to undo the effects of appserversetupuddi.jacl | UDDIReg/scripts |
| appserversetupuddi.jacl | Admin script to create a JDBC driver and datasource for the UDDI Registry, and to install the UDDI Registry application in a single, standalone, application server | UDDIReg/scripts |
| SetupDB2UDDI.jar | The 'UDDI DB2 Setup Wizard', to create and pre-load the UDDI Registry database if DB2 is to be used as the persistence store | UDDIReg/scripts |
| UDDI20 (directory) | Cloudscape directory containing the UDDI Registry tables and pre-loaded data | bin |
| uddiejbclient.jar | Class library for use when writing an EJB client to access the UDDI Registry | UDDIReg/ejb |
| Various javadoc files | JAVADOC to describe the EJB interface to the UDDI Registry | UDDIReg/ejb/javadoc |

If you intend to run in a Deployment Manager Cell then complete the following task - **Installing the UDDI Registry into a deployment manager cell**

If you intend to run in a single WebSphere Application server, then complete the following task - **Installing the UDDI Registry into a single WebSphere Application Server**

What to do next

Continue with Configuring the UDDI Registry.

## Installing the UDDI Registry into a deployment manager cell

These instructions assume that the installation has been performed into a clean environment. If you are installing into an existing Deployment Manager cell, then steps 1 to 5 must be omitted, i.e. skip to bullet 6 immediately.

Steps for this task

1. Install the WebSphere Application Server for Network Deployment package, and select the UDDI Registry option under Web Services.
2. Install one or more base application servers which will form the cell of servers. One of these should be the application server in which you plan to run an instance of the UDDI Registry. You can run more than one instance of the UDDI Registry within a cell of servers: the UDDI Registry application name will be suffixed with the target node and server names to make it unique within the cell (See also ″ Advanced use of setupuddi.jacl″), but you can only run one UDDI instance within each application server.
3. Ensure that the target application server is stopped.
4. Run *startManager* (*startManager.sh* on Unix Platforms) on the deployment manager node to start the deployment manager.

   (On Unix platforms you must remember to run the db2profile script before issuing the *startServer.sh server_name* command. This script is located within the DB2 instance's home directory under SQLLIB and can be invoked, for example, by typing:

   ```
   ". /home/db2inst1/sqllib/db2profile"
   ```

   )
5. Run *addNode* (*addNode.sh* on Unix Platforms) on each of the base application server(s) to add it as a node into the cell (please see elsewhere in the InfoCenter for how to run addNode). For example, addnode cell_host
6. Copy the *uddiejbclient.jar* file and the EJB javadoc directory tree from the UDDIReg/ejb subdirectory of the deployment manager install tree onto any machine(s) where you will be creating EJB clients to access the UDDI Registry.
7. If you have any global configuration properties that will be common to any UDDI Registries that you install into this cell, then you can edit the *uddi.properties* file in the properties subdirectory of the deployment manager install tree to set them up (see the section on Configuring the UDDI Registry for more details about the global configuration properties).
8. The UDDI Registry application is supplied with the security permissions that it requires to execute. This step explains how you can see the permissions that have been set, and change them if you feel that it is appropriate to do so. ***It is recommended that you only do this if you have a thorough understanding of Java 2 security issues, and the way in which security permissions are used by WebSphere.***

   The permissions for the UDDI Registry application are set within a file *was.policy* which is part of the *uddi.ear* application file. To see and change the contents of this file you should:

   a. On the deployment manager, copy the **uddi.ear** file from the installableApps subdirectory of the deployment manager install tree into a temporary directory.
   b. Un-jar the *uddi.ear* file (i.e. unpack uddi.ear using the 'jar -x' command).

      For example,

      ```
      jar -x uddi.ear
      ```

      (This uses the jar command in the bin subdirectory of the deployment manager, so you might need to fully qualify the path to the jar command.)
   c. You will find the *was.policy* file under the META-INF subdirectory that is created.

This will allow you to see the permissions which have been granted to the UDDI Registry application, and to make any changes that are necessary. Please note that if you make any errors in changing this file, then the UDDI Registry application might either fail to start, or will encounter errors when trying to execute UDDI requests.

d. Re-jar the *uddi.ear* file using the jar command.

   For example,

   ```
   jar -cf uddi.ear.
   ```

   **Note:** NOTE the dot after uddi.ear)

   (This uses the jar command in the bin subdirectory of the deployment manager, so you might need to fully qualify the path to the jar command.)

e. Copy the new *uddi.ear* back to the installableApps directory.

9. *Please note that if the target application server is running, then this step will stop and restart it.*If you are planning to use Cloudscape for the database in which the UDDI Registry will be held, please read the section ″Setting up the UDDI Registry to use Cloudscape within a deployment manager cell″ and then return to this point. If however, you plan to use DB2, then please refer to the section ″Setting up the UDDI Registry to use DB2 within a deployment manager cell″ and then return to this point.
10. Ensure that the UDDI Registry is configured appropriately for your installation, as described in the section on Configuring the UDDI Registry.
11. Start, or stop and restart, the target application server. This should also start the UDDI Registry application. If not, use the admin console on the deployment manager to do so.

What to do next

**Advanced use of setupuddi.jacl**

A number of symbols are defined at the top of the setupuddi.jacl script. These allow you to control the amount of logging that is performed, and to install multiple instances of the UDDI Registry within the same cell.

The symbols which you can edit are as follows:

- **logEnabled** - default setting is 1 which causes the progress of the script to be logged. Setting this symbol to 0 causes information logging to be suppressed, with only error messages being output.
- **overwriteExisting** - default setting is 1 which causes any existing installation of the UDDI Registry application to be overwritten. Setting this symbol to 0 would cause the existing installation to be left as is, but would allow other files used by the UDDI Registry to be updated. You are recommended to only change this setting under the guidance of IBM Service.
- **appName** - default setting is UDDIRegistry, which will be the first part of the name used for the UDDI Registry application installed into the target server. To ensure uniqueness of application names within the cell, the full application name that will be used is <appName>.<nodeName>.<server>, where <nodeName> is the name of the target node and <server> is the name of the target server. You can choose to change the first part of this (the <appName>) portion by changing the setting of this symbol before running setupuddi.jacl, although it is generally recommended that you do not change this value.

Continue with Configuring the UDDI Registry.

## Setting up the UDDI Registry to use Cloudscape within a deployment manager cell

If you plan to use Cloudscape for the database in which the UDDI Registry data will be held, use this task to create and load the UDDI Registry database using Cloudscape.

Before you begin

Please see the section ″Choice of database product to be used as the persistence store″ to decide which database product you should use as your persistence store before proceeding further with this step.

This task, to configure Cloudscape for the UDDI Registry database, is part of the parent task to install and setup a UDDI Registry, described in Installing the UDDI Registry into a deployment manager cell. You should complete this task at the appropriate step in the parent task, then return to and complete the parent task.

If you plan to use Cloudscape for the database in which the UDDI Registry data will be held, use this task to setup and install the UDDI Registry application to use the supplied Cloudscape database.

This task configures Cloudscape on the host where you want to run the UDDI Registry. Cloudscape is supplied with WebSphere Application Server, so you should not need to install Cloudscape support.

In this task you will invoke a script called setupuddi.jacl, specifying the target node and application server into which the UDDI Registry is to be deployed. Please note that if the target application server is running when you invoke setupuddi.jacl, then the script will stop the server and will restart the server after it has completed its operations.

Steps for this task

1. Copy the UDDI20 directory tree from the bin subdirectory of the deployment manager tree into the bin subdirectory of the target application server's install tree.
2. Create a JDBC driver and datasource to provide access to the UDDI20 Cloudscape database, and install the UDDI Registry application. This is done using the WSADMIN tool, using as input the setupuddi.jacl script from the UDDIReg/scripts subdirectory of the Deployment Manager. Note that this script must be run on the deployment manager node.

   You should either run this script from the UDDIReg/scripts subdirectory where it is located, or copy it to some other suitable directory. Note that the WSADMIN command is located in the bin subdirectory of the deployment manager node.The syntax for calling this script for Cloudscape is:

   ```
   wsadmin -f setupuddi.jacl
                deploymgrpath
                servername
                nodename
                discoveryURLprefix
                pathtodb
                > setupuddi.log
   ```

   where

   - *deploymgrpath* is the fully qualified pathname of the deployment manager install directory, specified using forward slashes regardless of platform; e.g. for Windows, this might be C:/Progra~1/WebSphere/deploymentManager

- *servername* is the name of the target server on which you wish to deploy the UDDI Registry, such as server1
- *nodename* is the name of the WebSphere node on which the target server runs
- *discoveryURLprefix* is the URL prefix to be used for discovery URLs. Typically this will be of the form http://<ip-address>:9080/uddisoap/ so an example of a discoveryURLprefix value might be http://mynode.mylocation.mycompany.com:9080/uddisoap/
- *pathtodb* is the path to the UDDI20 database within the bin subdirectory of your WebSphere AppServer installation, specified using forward slashes regardless of platform; e.g. for Windows, this might be C:/Progra~1/WebSphere/AppServer/bin/UDDI20 and for Unix platforms, it might be /opt/WebSphere/AppServer/bin/UDDI20
- > *setupuddi.log* is an optional parameter to direct the output to a log file as opposed to the default (which is to the screen)

For example, the following command (shown split for publication):

```
wsadmin -f setupuddi.jacl "C:/Progra~1/WebSphere/DeploymentManager/"
 server1 MYRIAD "http://myriad.headoffice.xyz.com:9080/uddisoap/"
 "C:/Progra~1/WebSphere/Appserver/bin/UDDI20"
```

will install the UDDI Registry application into the server server1 running on node MYRIAD, and set it up to access the Cloudscape UDDI20 database located in the bin subdirectory of the application server.

The setupuddi.jacl script will

a. Create a JDBC driver named UDDI.JDBC.Driver.<nodeName>.<server> and a datasource named UDDI.Datasource.<nodeName>.<server> (where <nodeName> is the name of the target node and <server> is the name of the target server>, and will replace any existing driver and datasource of that name.
b. Check whether the UDDI Registry application is already installed and, if so, stop it and uninstall it.
c. Update the uddi.properties configuration property file to configure the discoveryURLprefix value that you have specified and set the persister property as 'Cloudscape', and place this file into the location config/cells/<currentcell>/nodes/<nodename>/servers/<servername> /uddi.properties. You should make any further global configuration changes using this copy of the file.
d. Place a number of files that are needed by the UDDI Registry into the WebSphere configuration repository, and update the ws.ext.dirs list to reference these files.
e. Install the UDDI Registry.

This script will deploy the UDDI Registry into the configuration under the deployment manager, and then do a Synch which causes it to get installed into the specified server.

## What to do next

Return to the next step in the parent task Installing the UDDI Registry into a deployment manager cell.

## Setting up the UDDI Registry to use DB2 within a deployment manager cell

Before you begin

Please see the section ″Choice of database product to be used as the persistence store″ to decide which database product you should use as your persistence store before proceeding further with this step.

This task, to set up the UDDI Registry to use DB2 within a deployment cell, is part of the parent task to install the UDDI Registry into a deployment manager cell, described in Installing the UDDI Registry into a deployment manager cell. You should complete this task at the appropriate step in the parent task, then return to and complete the parent task.

If you plan to use DB2 for the database in which the UDDI Registry data will be held, use this task to create and load the UDDI Registry database using DB2, and to setup and install the UDDI Registry application to use the DB2 database.

This task uses the UDDI DB2 Setup Wizard to configure DB2 on the system where you want to run the UDDI Registry. Before starting this task, ensure that DB2 is installed and running on that system.

Copy the UDDIReg directory tree from the deployment manager to the target application server where DB2 will run.

The following steps should be carried out on the system on which the target application server is located (referred to below as the ′target system′).

In this task you will invoke a script called setupuddi.jacl, specifying the target node and application server into which the UDDI Registry is to be deployed. Please note that if the target application server is running when you invoke setupuddi.jacl, then the script will stop the server and will restart the server after it has completed its operations.

Steps for this task

1. On Windows, ensure that since installing DB2 you have run the usejdbc2.bat command file.

   For more information about this, see ″Application Building Guide″ in the DB2 documentation.
2. Create and load the UDDI Registry database, called UDDI20.

   **Note:** If you are migrating from an earlier version of the UDDI Registry, and your UDDI20 DB2 database already exists, then you should skip this step unless you want to replace the existing database with a new UDDI20 DB2 database. If you do choose to replace an existing database then all of your existing UDDI data will be lost. **Important:** please also note that if you do choose to replace the existing UDDI20 database, then there must not be any applications or users accessing the database at the time that you run the UDDI DB2 setup wizard.

   The UDDI DB2 Setup Wizard used in this task will prompt you to provide the DB2 userid and password under which the UDDI Registry database will be created and subsequently accessed. Before starting this task, ensure that you have created an appropriate DB2 userid and password. This same userid and password must be used throughout the following steps where the DB2 userid and password is requested.

On Windows this should be a userid and password with administrative privileges. On Unix platforms, you should supply the userid and password of the DB2 instance in which you wish the database to be created.

To create the database you use the UDDI DB2 setup wizard, which is supplied as a jar file called SetupDB2UDDI.jar in the UDDIReg/scripts subdirectory, by following these steps:

a. Change directory to the directory containing the file SetupDB2UDDI.jar (that is, either the UDDIReg/scripts directory in which it is supplied, or a directory on the target system into which you have copied it).
b. In order to run the wizard, you need to first ensure that you have access from your command line to the JVM supplied with WebSphere. This is done as follows:
   - On Windows, in a command window type the following command:
     ```
     was_install\bin\setupcmdline.bat
     ```
     Where *was_install* is the path to the directory where you installed WebSphere Application Server.

     For example,
     ```
     C:\Program Files\WebSphere\AppServer\bin\setupcmdline.bat
     ```
   - On Unix platforms, at a command line type one of the following commands:
     – If you are using bash, then as the root user run
     ```
      ./opt/WebSphere/AppServer/bin/setupCmdLine.sh
     ```
     – If you are using csh, then as the root user run
     ```
     source /opt/WebSphere/AppServer/bin/setupCmdLine.sh
     ```
     where */opt/WebSphere/AppServer* is the path to the directory where you installed WebSphere Application Server.
c. In the same command window, start the UDDI DB2 setup wizard by typing one of the following commands:
   - To start a graphical user interface, type
     ```
     java -jar SetupDB2UDDI.jar
     ```
   - To start a text mode interface, type:
     ```
     java -jar SetupDB2UDDI.jar -console
     ```
d. Follow the prompts to work through the wizard panels.
e. **(Optional)** If necessary, check the log files for the wizard. A log file called UDDIloadDB.log is written to the directory from which the wizard is run (but note that on Windows platforms, if you have decided not to overwrite an existing UDDI20 database, then this fact will not be logged, and the log file will not be created).

3. Create a JBDC driver and datasource to provide access to the UDDI20 DB2 database, and install the UDDI Registry application. This is done using the WASADMIN tool, using as input the setupuddi.jacl script from the UDDIReg/scripts subdirectory of the Deployment Manager. Note that this script must be run on the deployment manager node.

   You should either run this script from the UDDIReg/scripts subdirectory where it is located, or copy it to some other suitable directory. Note that the WSADMIN command is located in the bin subdirectory of the deployment manager node.The syntax for this script for DB2 is:

```
wsadmin -f setupuddi.jacl
             deploymgrpath
             servername
             nodename
             discoveryURLprefix
             dbname
             db2userid
             db2password
             db2ziplocation
             > setupuddi.log
```

where

- *deploymgrpath* is the fully qualified pathname of the deployment manager install directory, specified using forward slashes regardless of platform; e.g. for Windows, this might be c:/Progra~1/WebSphere/DeploymentManager
- *servername* is the name of the target application server on which you wish to deploy the UDDI Registry, such as server1
- *nodename* is the name of the WebSphere node on which the target application server runs. Typically, this will be the same as the machine name.
- *discoveryURLprefix* is the URL prefix to be used for discovery URLs. Typically this will be of the form http://<ip-address>:9080/uddisoap/ so an example of a discoveryURLprefix value might be http://mynode.mylocation.mycompany.com:9080/uddisoap/
- *dbname* is the name of the UDDI Registry database under DB2. You should specify UDDI20 for this parameter
- *db2userid* and *db2password* are a valid DB2 userid and password with administrative privileges
- *db2ziplocation* is the path to the db2java zip file on your system, specified using forward slashes regardless of platform; e.g. for Windows, this might be C:/Progra~1/SQLLIB/java/db2java.zip
- > *setupuddi.log* is an optional parameter to direct the output to a log file as opposed to the default (which is to the screen)

For example (shown split for publication):

```
wsadmin -f setuppuddi.jacl "C:/Progra~1/WebSphere/deploymentManager/"
 server1 MYRIAD "http://myriad.headoffice.xyz.com:9080/uddisoap/"
 UDDI20 db2admin secretpwd "C:/Progra~1/SQLLIB/java/db2java.zip" >
 setupuddi.log
```

will install the UDDI Registry application into the server server1 running on node MYRIAD, and set it up to access the DB2 UDDI20 database using the userid 'db2admin' and password 'secretpwd'.

The setupuddi.jacl script will:

a. Create a JDBC driver named UDDI.JDBC.Driver.<nodeName>.<server> and a datasource named UDDI.Datasource.<nodeName>.<server> (where <nodeName> is the name of the target node and <server> is the name of the target server>, and will replace any existing driver and datasource of that name.
b. Check whether the UDDI Registry application is already installed and, if so, stop it and uninstall it.
c. Update the uddi.properties configuration file to configure the discoveryURLprefix value that you have specified, and to set the persister property as 'DB2', and place this file into the location

config/cells/<currentcell>/nodes/<nodename>/servers/<servername> /uddi.properties. You should make any further global configuration changes using this copy of the file.

d. Place a number of files that are needed by the UDDI Registry into the WebSphere configuration repository, and update the ws.ext.dirs list to reference these files.
e. Install the UDDI Registry.

What to do next

Return to the next step in the parent task Installing the up a UDDI Registry into a deployment manager cell.

## Installing the UDDI Registry into a single appserver

If you intend to run in a single WebSphere Application server, then complete the following task.

When you select the UDDI Registry option, then the installation will place all files that are needed to run a UDDI Registry onto the deployment manager install tree on the machine on which you are installing IBM WebSphere Application Server for Network Deployment.

To be able to run the UDDI Registry in a single application server instance in your network space you must copy these files over to the application server and then deploy the UDDI Registry. You can do this as follows:

Steps for this task

1. Stop the application server on which you plan to run the UDDI Registry; for example, using the command `stopServer server_name` (*stopServer.sh* on Unix Platforms)
2. Copy the *uddi.ear* file from the installableApps subdirectory of the deployment manager install tree into the installableApps subdirectory of the target application server's install tree.
3. Copy the *uddi.properties* file from the properties subdirectory of the deployment manager install tree into the properties subdirectory of the target application server's install tree.

   In a subsequent step, you configure the UDDI Registry using the properties in the *uddi.properties* file.
4. Copy both the *uddiresourcebundles.jar* and the *setupuddimessages.jar*files from the lib subdirectory of the deployment manager install tree into the lib subdirectory of the target application server's install tree.
5. Copy the *uddiejbclient.jar* file and the EJB javadoc directory tree from the UDDIReg/ejb subdirectory of the deployment manager install tree onto any machines where you will be creating EJB clients to access the UDDI Registry.
6. Configure database support for the UDDI Registry database, in which the UDDI Registry will be held.

   To do this, complete one of the following tasks then return this point:
   - Setting up the UDDI Registry to use Cloudscape in a single AppServer
   - Setting up the UDDI Registry to use DB2 in a single AppServer

   **Note:** If you set up the UDDI Registry application with a JDBC driver and datasource that reference Cloudscape, but set the persister property in *uddi.properties* to specify DB2, **or vice versa**, then some unexpected behavior will result, such as a fatal error on deleting an entity. If this happens, you should check that the above details are not in conflict. This only applies to a UDDI Registry installation on a single appserver.

7. Ensure that the UDDI Registry is configured appropriately for your installation, as described in the section on Configuring the UDDI Registry.
8. Stop then restart the application server.

   (On Unix platforms you must remember to run the db2profile script before issuing the *startServer.sh server_name* command. This script is located within the DB2 instance's home directory under SQLLIB and can be invoked, for example, by typing:

   ```
   ". /home/db2inst1/sqllib/db2profile"
   ```

   )

What to do next

Continue with Configuring the UDDI Registry.

## Setting up the UDDI Registry to use Cloudscape in a single appserver

Before you begin

Please see the section ″Choice of database product to be used as the persistence store″ to decide which database product you should use as your persistence store before proceeding further with this step.

If you plan to use Cloudscape for the database in which the UDDI Registry data will be held, use this task to setup and install the UDDI Registry application to use the supplied Cloudscape database.

This task configures Cloudscape on the host where you want to run the UDDI Registry. Before starting this task, ensure that Cloudscape is installed and running on that host, and that you can open a command window on that host.

This task, to configure Cloudscape for the UDDI Registry database. Cloudscape is supplied with WebSphere Application Server, so you should not need to install Cloudscape support.

This task, to configure Cloudscape for the UDDI Registry database, is part of the parent task to install and setup a UDDI Registry, described in Installing and Setting up a UDDI Registry. You should complete this task at the appropriate step in the parent task, then return to and complete the parent task.

To configure Cloudscape for the UDDI Registry database, complete the following steps:

Steps for this task

1. Copy the UDDI20 directory tree from the bin subdirectory of the deployment manager tree into the bin subdirectory of the target application server's install tree.
2. Copy the uddicloudscapeuserfunc.jar file from the lib subdirectory of the deployment manager install tree to the lib subdirectory of the target application server's install tree.

3. Ensure that the persister property in the uddi.properties file is set to persister=Cloudscape
4. Copy the *appserversetupuddi.jacl* script from the UDDIReg/scripts subdirectory of the deployment manager install tree to the WebSphere Application Server bin subdirectory (for example, on Windows, C:\Progra~1\WebSphere\AppServer\bin).
5. Change directory to the WebSphere Application Server bin subdirectory.
6. Start the application server on which the UDDI Registry is to run.

   For example, you can start the application server server1 by typing the command:

   ```
   startserver server1
   ```
7. Create a JDBC driver and datasource to provide access to the UDDI20 Cloudscape database, and install the UDDI Registry application.

   To do this run the WSADMIN tool with the script *appserversetupuddi.jacl* as input, on the target application server, using the following command syntax:

   (You should either run this script from the UDDIReg/scripts subdirectory where it is located, or copy it to some other suitable directory. Note that the WSADMIN command is located in the WebSphere bin subdirectory.)

   ```
   wsadmin -f appserversetupuddi.jacl
           uddi-ear-location
           servername
           nodename
           WebSphere-lib-subdirectory
           cloudscapedbname
           > setupuddi.log
   ```

   Where:

   - *uddi-ear-location* is the fully-qualified path to the uddi.ear file in the installableApps subdirectory, specified using forward slashes regardless of platform. For example, on Windows:

     `C:/Progra~1/WebSphere/AppServer/installableApps/uddi.ear`
   - *servername* is the name of the application server on which the UDDI registry is to run; for example: `server1`
   - *nodename* is the name of the WebSphere node on which the application server, *servername*, is running. Typically this is the machine name.
   - *WebSphere-lib-subdirectory* is the fully-qualified path to the WebSphere Application Server lib subdirectory, specified using forward slashes regardless of platform. For example:
     - On Windows: `C:/Progra~1/WebSphere/AppServer/lib`
     - On Unix: `/opt/WebSphere/AppServer/lib`
   - *cloudscapedbname* is the fully-qualified path to the UDDI20 database within the bin subdirectory of your WebSphere AppServer installation, specified using forward slashes regardless of platform. For example:
     - On Windows, `C:/Progra~1/WebSphere/AppServer/bin/UDDI20`
     - On Unix, `/opt/WebSphere/AppServer/bin/UDDI20`
   - > *setupuddi.log* is an optional parameter to direct the output to a log file as opposed to the default (which is to the screen)

   The appserversetupuddi.jacl script will complete the following actions:

   a. Create a JDBC driver named UDDI.JDBC.Driver.<nodeName>.<server> and a datasource named UDDI.Datasource.<nodeName>.<server> (where

&lt;nodeName&gt; is the name of the target node and &lt;server&gt; is the name of the target server&gt;, and will replace any existing driver and datasource of that name.

b. Checks whether the WebSphere UDDI Registry application is already installed and, if so, stop the application and uninstall it.

c. Installs the WebSphere UDDI Registry, then start it.

What to do next

Return to the next step in the parent task Installing the UDDI Registry into a single appserver.

## Setting up the UDDI Registry to use DB2 in a single appserver

Before you begin

Please see the section ″Choice of database product to be used as the persistence store″ to decide which database product you should use as your persistence store before proceeding further with this step.

This task, to configure DB2 for the UDDI Registry database, is part of the parent task to install and setup a UDDI Registry, described in Installing and setting up a UDDI Registry. You should complete this task at the appropriate step in the parent task, then return to and complete the parent task.

If you plan to use DB2 for the database in which the UDDI Registry data will be held, use this task to create and load the UDDI Registry database using DB2, and to setup and install the UDDI Registry application to use the database.

This task uses the UDDI DB2 setup wizard to configure DB2 on the system where you want to run the UDDI Registry. Before starting this task, ensure that DB2 is installed and running on that system.

Copy the UDDIReg directory tree from the deployment manager to the target application server where DB2 will run.

The following steps should be carried out on the system on which the target application server is located (referred to below as the ′target system′).

Steps for this task

1. On Windows, ensure that since installing DB2 you have run the usejdbc2.bat command file.

   For more information about this, see ″Application Building Guide″ in the DB2 documentation.

2. Create and load the UDDI Registry database, called UDDI20.

   **Note:** If you are migrating from an earlier version of the UDDI Registry, and your UDDI20 DB2 database already exists, then you should skip this step unless you want to replace the existing database with a new UDDI20 DB2 database. If you do choose to replace an existing database then all of your existing UDDI data will be lost. **Important:** please also note that if you do choose to replace the existing UDDI20 database, then there must not be any applications or users accessing the database at the time that you run the UDDI DB2 setup wizard.

   The UDDI DB2 setup wizard used in this task will prompt you to provide the DB2 userid and password under which the UDDI Registry database will be created and subsequently accessed. Before starting this task, ensure that you have created an appropriate DB2 userid and password. This same userid and password must be used throughout the following steps where the DB2 userid and password is requested.

On Windows this should be a userid and password with administrative privileges.

On Unix platforms, you should supply the userid and password of the DB2 instance in which you wish the database to be created.

To create the database you use the UDDI DB2 setup wizard, which is supplied as a jar file called SetupDB2UDDI.jar in the UDDIReg/scripts subdirectory, by following these steps:

a. Change directory to the directory containing the file SetupDB2UDDI.jar (that is, either the UDDIReg/scripts directory in which it is supplied, or a directory on the target system into which you have copied it).
b. In order to run the wizard, you need to first ensure that you have access from your command line to the JVM supplied with WebSphere. This is done as follows:
   - On Windows, in a command window type the following command:
     ```
     was_install\bin\setupcmdline.bat
     ```
     Where *was_install* is the path to the directory where you installed WebSphere Application Server.

     For example, C:\Program Files\WebSphere\AppServer\bin\setupcmdline.bat
   - On Unix platforms, at a command line type one of the following commands:
     – If you are using bash, then as the root user run
       ```
       . /opt/WebSphere/AppServer/bin/setupCmdLine.sh
       ```
     – If you are using csh, then as the root user run
       ```
       source /opt/WebSphere/AppServer/bin/setupCmdLine.sh
       ```
     where */opt/WebSphere/AppServer* is the path to the directory where you installed WebSphere Application Server.
c. In the same command window, start the UDDI DB2 setup wizard by typing one of the following commands:
   - To start a graphical user interface, type
     ```
     java -jar SetupDB2UDDI.jar
     ```
   - To start a text mode interface, type:
     ```
     java -jar SetupDB2UDDI.jar -console
     ```
d. Follow the prompts to work through the wizard panels.
e. **(Optional)** If necessary, check the log files for the wizard. A log file called UDDIloadDB.log is written out into the directory from which the wizard is run (but note that, on Windows platforms, if you have decided not to overwrite an existing UDDI20 database, then this fact will not be logged, and the log file will not be created).

3. Ensure that the persister property in the uddi.properties file is set to `persister=DB2`.
4. On Unix, ensure that you have run the db2profile script to set up the environment for the DB2 instance that the UDDI Registry is using.

   For example, type the command:

```
. /home/db2inst1/sqllib/db2profile
```

5. Start the application server on which the UDDI Registry is to run.

   For example, you can start the application server server1 by typing the command:

   ```
   startserver server1
   ```

6. Copy the *appserversetupuddi.jacl* script from the UDDIReg/scripts subdirectory of the deployment manager install tree to the WebSphere Application Server bin subdirectory (for example, on Windows, C:\Progra~1\WebSphere\AppServer\bin).
7. Change directory to the WebSphere Application Server bin subdirectory.
8. Create a JDBC driver and datasource to provide access to the UDDI20 database, and install the UDDI Registry application.

   To do this run the WSADMIN tool with the script *appserversetupuddi.jacl* as input, on the target application server, using the following command syntax:

   (You should either run this script from the UDDIReg/scripts subdirectory where it is located, or copy it to some other suitable directory. Note that the WSADMIN command is located in the WebSphere bin subdirectory.)

   ```
   wsadmin -f appserversetupuddi.jacl
               uddi-ear-location
               servername
               nodename
               WebSphere-lib-subdirectory
               dbname
               db2userid
               db2pwd
               db2-install-dir/java12/db2java.zip
               > setupuddi.log
   ```

   where

   - *uddi-ear-location* is the fully-qualified path to the uddi.ear file in the installableApps subdirectory, specified using forward slashes regardless of platform. For example, on Windows:

     `C:/Progra~1/WebSphere/AppServer/installableApps/uddi.ear`
   - *servername* is the name of the application server on which the UDDI registry is to run; for example: `server1`
   - *nodename* is the name of the WebSphere node on which the application server, *servername*, is running
   - *WebSphere-lib-subdirectory* is the fully-qualified path to the WebSphere Application Server lib subdirectory, specified using forward slashes regardless of platform. For example:
     - On Windows: `C:/Progra~1/WebSphere/AppServer/lib`
     - On Unix: `/opt/WebSphere/AppServer/lib`
   - *dbname* is the name of the UDDI Registry database under DB2. You should specify UDDI20 for this parameter
   - *db2userid* and *db2pwd* are a valid DB2 userid and password with administrative privileges, as specified in an earlier step.
   - *db2-install-dir* is the path under which you have installed DB2 on your system, specified using forward slashes regardless of platform. For example, on Windows: `C:/Progra~1/SQLLIB`
   - > *setupuddi.log* is an optional parameter to direct the output to a log file as opposed to the default (which is to the screen)

   The appserversetupuddi.jacl will complete the following actions:

a. Create a JDBC driver named UDDI.JDBC.Driver.<nodeName>.<server> and a datasource named UDDI.Datasource.<nodeName>.<server> (where <nodeName> is the name of the target node and <server> is the name of the target server>, and will replace any existing driver and datasource of that name.
b. Checks whether the WebSphere UDDI Registry application is already installed and, if so, stop the application and uninstall it.
c. Installs the WebSphere UDDI Registry, then starts it.

What to do next

Return to the next step in the parent task Installing and Setting up a UDDI Registry.

## Reinstalling the UDDI Registry application

If you wish to reinstall the UDDI Registry then please follow the appropriate section below.

**Reinstalling into a deployment manager cell**

If you wish to reinstall the UDDI Registry into the target application server, for example because you wish to alter certain aspects of its configuration using AAT, then you should rerun the setupuddi.jacl script (described in the appropriate link below

- ″Setting up the UDDI Registry to use Cloudscape within a deployment cell″
- ″Setting up the UDDI Registry to use DB2 within a deployment cell″

**Reinstalling into a single appserver**

Remove the UDDI Registry application in the same manner as any other Enterprise Application and then install using the appropriate link shown below:

- Setting up the UDDI Registry to use Cloudscape in a single AppServer
- Setting up the UDDI Registry to use DB2 in a single AppServer

## Removing the UDDI Registry application from a deployment manager cell

Before you begin

If you wish to completely remove the UDDI Registry application from the target application server in the deployment manager cell, then you should run the *wsadmin* (*wsadmin.sh* on Unix Platforms) script *removeuddi.jacl*, which is located in the UDDIReg/scripts directory of the deployment manager install tree.

Please note that if the target server specified on invoking removeuddi.jacl is running at the same time, then the script will stop the server and will restart the server after when it has completed its operations.

Steps for this task

1. The syntax for this script is:

```
wsadmin -f removeuddi.jacl
            servername
            nodename
            > removeuddi.log
```

Where *servername* and *nodename* are the server and node where you have deployed the UDDI Registry application. By default output will go to the screen, but, optionally, you can specify '> removeuddi.log' to direct output to a log file.

For example,

```
wsadmin -f removeuddi.jacl server1 myriad
```

will remove the UDDI Registry application and related files from server server1 running in node myriad, and will send any messages to the screen.

## Removing the UDDI Registry application from a single appserver

Before you begin

If you wish to completely remove the UDDI Registry application from a standalone application server then you should run the WSADMIN script appserverremoveuddi.jacl, which will have been installed into the UDDIReg/scripts directory when you installed the UDDI Registry as part of a Network Deployment install.

Steps for this task are:

Steps for this task

1. The syntax for this script is:

   ```
   wsadmin -f appserverremoveuddi.jacl
                 servername
                 nodename
                 > removeuddi.log
   ```

   where

   - *servername* and *nodename* are the name of the standalone application node in which it runs (these are the names that you specified when you ran appserversetupuddi.jacl to install the UDDI Registry application).
   - by default output will go to the screen, but, optionally, you can specify '> *removeuddi.log*' to direct the output to a log file.

   For example,

   ```
   wsadmin -f appserverremoveuddi.jacl server1 monolith
   ```

   will remove the UDDI Registry application and related files from server server1 running in node monolith, and will send any messages to the screen.

## Configuring the UDDI Registry

Before you begin

The UDDI Registry is supplied as a J2EE application file, uddi.ear. This is installed into the WebSphere Application Server during installation. If you want to change any of its configuration properties using AAT see "Configuring SOAP properties with the AAT".

If you enable WebSphere security then to run the publish API servlet of the IBM WebSphere UDDI Registry, you also need to configure WebSphere to use HTTPS and SSL, as described in Configuring WebSphere to use HTTPS and SSL

You can configure the following aspects of the UDDI Registry:
- ″Configuring global UDDI properties″
- ″Modifying the database userid and password″
- ″Configuring security properties″
- ″Configuring the UDDI User Console (GUI) for multiple language encoding support″
- ″Customizing the UDDI User Console (GUI)″
- ″Configuring SOAP interface properties″
- ″Configuring SOAP properties with the AAT″
- ″Configuring SOAP properties in the deployment descriptor″
- ″Configuring WebSphere to use HTTPS and SSL″

## Configuring global UDDI properties

Before you begin

To set the following global UDDI properties, edit the properties file *uddi.properties*, which is in one of the following locations depending on where you have installed the UDDI Registry:
- If you have installed the UDDI Registry into an application server within a Deployment Manager cell, then the uddi.properties file will be located in the configuration repository, under the application server; that is in config/cells/<cellname>/nodes/<nodename>/servers/<servername>, where <cellname> is the name of the Deployment Manager cell, <nodename> is the name of the node in which the application server is installed, and <servername> is the name of the application server in which you have installed the UDDI Registry.
- If you have installed the UDDI Registry into a single application server which is not part of a Deployment Manager cell, then the uddi.properties file will be located in the properties subdirectory of the WebSphere Application Server in which you have installed the UDDI Registry application.

The properties that can be changed within uddi.properties are as follows:
- The dbMaxResultCount, which is the limit on the number of rows of information that should be returned on Find requests, and will apply if the request does not specify a maxRows limit itself (or if it specifies a limit which exceeds this value). The initial value for this in *uddi.properties* is 100.
- The persister, which indicates what database is to be used as the persistence store for the UDDI Registry database. If you have installed the UDDI Registry into an application server within a Deployment Manager cell, then the persister property will have been set to the corect valuefor you. If you change this value, then you must also ensure that you have a UDDI Registry database created using the chosen database product (for more details about the UDDI Registry database, please refer to the section on ″Installing the UDDI Registry″). You should also be aware that any data published to the UDDI Registry with one setting of the persister property will **not** be accessible when running the UDDI Registry application with a different setting for the persister property. The valid values for the persister property are:
  – persister=DB2

    indicating that DB2 is to be used as the persistence store
  – persister=Cloudscape
    indicating that Cloudscape is to be used as the persistence store

  The initial value for this in *uddi.properties* is Cloudscape.

  **Note:** This property is dynamically set by the setupuddi.jacl script when installing into a Deployment Manager cell so in this case you should not need to modify it.
- The default language to be used on a publish request as the xml:lang when one is not specified. The initial value for this in *uddi.properties* is en-US. This property must contain one of the valid xml:lang values.
- The UDDI site operator name. This is a string which is stored in every registry object, to indicate the operator of the UDDI Registry. The initial value for this in *uddi.properties* is `www.mycompany.com/uddi`. This property does not have any particular functional use, so its value can be set to any string that you feel is suitable.
- The maximum number of search keys that can be used on find API requests. The initial value for this in *`uddi.properties`* is 5.
- The getServletURLprefix and getServletname name, used to build up the discovery URL. The initial values for these are `http://localhost:9080/uddisoap/` and `get`. If you have installed the UDDI Registry into an application server within a Deployment Manager cell, then the getServletURLPrefix property will have been set for you using the value you specified as a parameter to the setup script. You are recommended to set suitable values for these properties before you first use the UDDI Registry.

  **Note:** This property is dynamically set by the *setupuddi.jacl* script when installing into a Deployment Manager cell so in this case you should not need to modify it.

In order for your changes to take effect, you must stop and restart the UDDI Registry application using the WebSphere Administrative Console.

## Modifying the database userid and password

Before you begin

If you use DB2 as the persistence store for the UDDI Registry, and you need to change the database userid and/or password, you should alter the user and password values in the custom properties of the 'UDDI Datasource', which can be edited from the WebSphere Administrative Console. The UDDI.Datasource is under datasources within the UDDI.JDBC.Driver, which is itself found under JDBC Providers under Resources. Do not alter the databaseName.

## Configuring security properties

Before you begin

See Configuring WebSphere to use HTTPS and SSL for details on configuring security properties.

To run the publish API servlet of the IBM WebSphere UDDI Registry, you also need to configure WebSphere to use HTTPS and SSL, as described in Configuring WebSphere to use HTTPS and SSL

## Configuring the UDDI User Console (GUI) for multiple language encoding support

Before you begin

If you want to use multiple language encoding support in the User Console (GUI), you need to configure the application server into which the UDDI Registry application is installed with UTF-8 encoding enabled. To do this, please refer to ("Configuring application servers for UTF-8 encoding") elsewhere in the WebSphere Application Server version 5 InfoCenter on enabling an application server for UTF-8.

## Customizing the UDDI User Console (GUI)

Before you begin

The look and feel of the UDDI Console is determined by the styles defined in the *uddi_gui.css* file which is located in the /gui.war/theme directory of the installed UDDI Registry application directory. The UDDI Registry The UDDI Registry application directory will be one of the following, depending on where you ahve installed the UDDI Registry:

- If you have installed the UDDI Registry into an application server within a Deployment Manager cell, the directory is UDDIReg.ear under the /apps directory of the configuration repository for the cell.
- If you have installed the UDDI Registry into a single application server which is not part of a Deployment Manager cell, the directory is UDDIRgistry.ear under the installedApps directory of the WebSphere Application Server in which you have installed the UDDI registry application.

The contents of this file can be edited to change the colors, fonts and font sizes according to the user's preference.

The content and layout of the UDDI User Console is provided by JSP pages, which can be customized by a programmer who is familiar with JSPs. The JSP pages are found in the uddi.ear enterprise application, which is under the installedApps subdirectory of the WebSphere AppServer installation. To locate the JSPs, you should expand the UDDI_Registry.ear, open the gui.war, and they are located under WEB-INF in the pages subdirectory. So, on a Windows system which has WebSphere installed in the default location, the JSP files will be found in "<WebSphere-install-dir>\installedApps\UDDI_Registry.ear\gui.war\WEB-INF\pages". These JSP pages also contain some application logic (as opposed to presentation logic) which should not be changed.

## Configuring SOAP interface properties

Before you begin

You can configure the following SOAP interface properties

- *defaultPoolSize* - the number of SOAP parsers with which to initialize the parser pool for the SOAP interface. The can be set independently for the Publish (uddipublish) and Inquiry (uddi) APIs. For example, if you expect more inquiries than publish requests via the SOAP interface, you can set a larger pool size for the Inquiry API. The default initial size for both APIs is 10.
- The *context root* used for the Publish and Inquiry APIs, which forms a part of the URL by which they are accessed. By default this is /uddisoap.

- Whether the API is to be secure (via HTTPS) or insecure (via HTTP). The default is to use HTTPS.

To configure the following SOAP interface properties, you can use either of the following methods, as described below:

- ″Configuring SOAP properties with the AAT″ (the recommended option, especially for a production environment)
- ″Configuring SOAP properties in the deployment descriptor″ for the SOAP module in the UDDI application directly. This option is faster and may be the preferred method in a test environment.

## Configuring SOAP properties with the AAT

Before you begin

To configure SOAP properties by using the WebSphere Application Assembly tool, complete the following steps:

- Select *Update* and click on the Application icon.
- Select the *uddi.ear* file (this is placed, by the UDDI installation, into the UDDI install directory (e.g. *C:\WebSphere\installableApps\uddi.ear*).
- Expand the *uddi.ear* icon on the left hand pane in the AAT.
- Expand the *Web Modules* tree.
- Expand the *uddi Soap* tree
- To change the *defaultPoolSize,* expand Web Components and then *uddipublish* (for the publish API) or *uddi* (for the inquiry API).
  - Click on *Initialization Parameters* which will show the *defaultPoolSize* parameter in the upper right hand pane. This can be edited in the lower right-hand pane.
- To change the *context root*, click on *UDDI Soap* which will display general information about the SOAP module in the lower right hand pane in AAT. The *context root* can be edited in this pane.
- To change the publish API to use HTTP (instead of HTTPS), click on *Security Constraints* and change the *Transport Guarantee* from Confidential to none.
- Having made any changes above, you must now save them. To do this, click on *File -> Save (or Save As)* to save your changes.
- Redeploy the *uddi.ear* to WebSphere, by first removing it and reinstalling it via the Administrator's Console.

## Configuring SOAP properties in the deployment descriptor

Before you begin

To configure SOAP properties by using the WebSphere Application Assembly tool, complete the following steps:

1. The deployment descriptor for the SOAP module (*web.xml)* is found in the *WEB-INF* subdirectory of the *uddi.ear* application in the installed applications within the WebSphere install directory (for example, <*WebSphere-install-dir*>*\installableApps\uddi.ear\soap.war\WEB-INF*). It can be edited directly to specify the desired settings.
2. Stop and restart the WebSphere Application server for the changes to take effect.

### Configuring WebSphere to use HTTPS and SSL

Before you begin

To support the use of secure access with the IBM WebSphere UDDI Registry, you need to configure WebSphere to use HTTPS and SSL. Please refer to the information available elsewhere in this InfoCenter for configuring SSL in WebSphere Application Server. It is assumed throughout the information for the UDDI Registry that, where SSL is used, it has been configured on port 9443.

## Administering the UDDI Registry

Before you begin

This section describes the various tasks about administering the UDDI Registry.

- ″Running the UDDI Registry″
- ″Backing up and restoring the UDDI Registry database″

### Running the UDDI Registry

Before you begin

The UDDI Registry is started automatically when the Application Server is started. In order to stop and restart it, use the Administrative console.

### Backing up and restoring the UDDI Registry database

Before you begin

If you want to protect the data in your UDDI Registry database, you can backup and restore the database using the facilities of the database product. For DB2, you can do this by using the export and import functions of the DB2 Control Center. For Cloudscape you can simply use operating system tools to copy the database directory. Please refer to the database product information for more details.

The UDDI Registry database is called UDDI20, and the tables which should be backed up are:

- ADDRESS
- ADDRLINE
- BSERVICE
- BTEMPLATE
- BUSINESS
- CATEGORY
- CATEGORYBAG
- CONTACT
- DESCR
- DISCOVERYURL
- EMAIL
- EXTCATEGORY
- IDENTIFIERBAG
- INSTANCEDETAIL
- NAMEELEMENT
- OVERVIEWDOC

- PHONE
- PUBLISHERASSERTION
- SERVICEPROJECTION
- TMODEL
- VALIDATIONCACHE
- VALIDATIONSERVICES

## UDDI user console

This topic describes the layout of the UDDI user console, also referred to as the Graphical User Interface (GUI), which you can use to act on the IBM WebSphere UDDI Registry.

For information about how to display the UDDI user console, see Displaying the user console.

If you will be using the UDDI Console, then it is recommended that you configure the application server into which you have installed the UDDI Registry for UTF-8 encoding support: please refer to the section on ″Configuring the UDDI User Console for multiple language encoding support″.

The UDDI user console is split into three distinct areas. At the top of the screen are buttons which activate various functions in the areas below this bar. These buttons are:

**Home** returns you to the IBM WebSphere UDDI Registry welcome page

**Find** activates the Find tab on the frame below to the left

**Publish**
similarly activates the Publish tab on the frame below to the left

Below the WebSphere UDDI Registry banner the screen is split into two parts. On the left are the two tabs mentioned above, the Find and Publish tabs.

**Find tab**

The Find tab is in two parts. At the top, a **Quick Find** service is provided. There are three radio buttons to enable a choice of ′service′, ′business′ and ′technical model′ finds. Below these radio buttons is a text entry box for entering the name to search for and, beneath this, a ′Find′ link to start the search. Comments are provided to show the user the wildcard character. The results of clicking on the ′Find′ link are shown in the detail frame to the right.

Beneath the Quick Find is a section for **Advanced Find** functions which enables the user to choose which entity they want to perform an advanced search on. There are three links: Find services, Find businesses and Find technical models. Clicking one of these links displays the corresponding advanced search form in the frame to the right, which the user may use to enter search criteria. The **Locator** section has a link (marked in blue with the words ″**Show category tree**″) which displays the tree from which the user can select categories (or taxonomies). This is shown in the left hand frame In the advanced search form there are two links to start the search (mid-way down and at the bottom).

The results of clicking either of the two links to start the search are displayed in the same detail frame.

**Publish tab**

The *Publish* link on the top banner activates the Publish tab in the navigation frame to the left.

The Publish tab is split into three distinct sections. The top part is a **Quick Publish** section to allow the user to publish a business or technical model by name only. There are two radio buttons to enable a choice of 'business' or 'technical model'. Below these radio buttons is a text entry box for entering the name to assign to the selected entity and, beneath this, a blue 'Publish now' link to publish the entity. The results of clicking on the **Publish now** link are shown in the detail frame to the right.

To publish an entity with more detail, such as with multiple names, descriptions and categories, use the **Advanced Publish** section below this. The comments below each link ('Add a business' and 'Add a technical model') describe individual functions. Clicking one of these links displays the corresponding advanced publish form in the detail frame where the user may enter details about the entity they want to publish. As in the Advanced Find functions above, there are two links to publish a business or technical model (one towards the top of the form and the other at the bottom). Similarly the **Locator** section allows taxonomies to be shown in the left frame from which the user can select categories.

Below the Advanced Publish section is a **Registered Information** section which has a link to **Show Owned Entities** in order to show the businesses, services and technical models registered to the individual user, and pending business relationships. Clicking the **Show Owned Entities** link displays the **Show Owned Entities** page in the detail frame at the right. The **Show Owned Entities** page is organised in three sections: **Registered Businesses**, **Pending Business Relationships** and **Registered Technical Models**. Each section shows the number of registered items. Users can **Edit** or **Delete** businesses owned by them by clicking the appropriate links in the **Actions** column.

Services are added to a business by clicking the **Add a Service** link in the **Services** column of the **Registered Businesses** section. Services can also be 'referenced' by a business as if the business was the owner of the service. This 'service projection' is performed by clicking the **Reference a service** link in the **Services** column. Services associated with a business, whether they are owned or referenced, can be displayed by clicking the **Show services** link. This acts as a toggle, displaying services available for editing or deleting.

A business can be associated with another business in the UDDI Registry and this function is performed by clicking the **Add a relationship** link in the **Actions** column of the Registered Businesses section. Clicking the **Show related businesses** link in the **Actions** column displays a list of any completed business relationships.

The **Pending Business Relationships** section shows all incomplete publisher assertions, where only one party has asserted a relationship and is waiting for the other party to make the same assertion. This section reminds the user of any relationships that involve their businesses. Once both parties have asserted the same relationship between two businesses, the relationship moves from the **Pending Business Relationships** section and appears in the list of relationships displayed after clicking the **Show related businesses** link in the **Registered Businesses** section.

Technical Models owned by the user are shown in the bottom **Registered Technical Models** section. As for businesses, users can Edit or Delete technical models owned by them by clicking the appropriate links in the **Actions** column.

**Note:** Users should take note that deletion of Technical Models (tModels) does **not** cause them to be physically deleted, but hidden. This is in accordance with the UDDI Registry V2.0 specifications. After deletion Technical Models are shown under the ″**Shown Owned Entities**″ link on the publish page but not via the Find links on the Find page. ALL other entities are deleted from the UDDI Registry in the normal way.

- ″Displaying the user console″

## Displaying the user console

Before you begin

This topic describes how to display the UDDI Registry user console (sometimes referred to as the GUI). The URL you use depends on whether or not you have enabled WebSphere security:

- If you have the WebSphere security disabled, you can access the UDDI User Console by using the following URL in your Web browser:

  ```
  http://<hostname>:9080/uddigui
  ```

  **Note:** With WebSphere security disabled, all the publish operations are performed using a userid of UNAUTHENTICATED.
- If you have WebSphere security enabled, you can access the UDDI User Console through HTTPS by using the following URL in your Web browser:

  ```
  https://<hostname>:9443/uddigui
  ```

  The User Console displays the default frameset containing the header frame, navigation frame showing find options, and details frame. When you click the link to show the publish options in the navigation frame, you are challenged for a userid and password.

  If WebSphere security is enabled and you try to access a publish action via an unsecured link, e.g. clicking the publish link on the navigation frame where the User Console was opened with

  ```
  http://<hostname>:9080/uddigui
  ```

  you will be redirected to a secure logon. Inquire functions will work as expected.

## SOAP Application Programming Interface for the UDDI Registry

Access to the SOAP API will by default be available at:

http://localhost:9080/uddisoap/inquiryapi or

https://localhost:9443/uddisoap/publishapi

Where ’localhost’ is the address by which your WebSphere server is known. If security is enabled on your WebSphere server, the publishapi will also be protected by basic-authentication. By default, when security is enabled, the publishapi is restricted to HTTPS, this is to ensure the confidentiality and security of your data whilst in transit to UDDI. If you do not wish to use SSL, when security is enabled, you will have to modify the jar file using AAT to remove the CONFIDENTIAL restriction placed upon the publish URLs. For more information about this topic, see the section on **Configuring SOAP properties with the AAT** If you normally access your WebSphere server via a web server, you will need to ensure the plugin configuration for the WebSphere plugin on the web server has been updated since installing UDDI. This will then allow access to the UDDI SOAP API through the URLs

http://localhost/uddisoap/inquiryapior

https://localhost/uddisoap/publishapi

Where 'localhost' is the address by which your web server is accessed. Note that if you plan on accessing UDDI via a web server in this manner, that the samples will require modification to inform them of the SSL certificates used by your web server, so that the samples can make SSL connections to the web server. It is beyond the scope of this document to cover the many variants available on web server/WebSphere/java SSL configurations

- ″Programming the UDDI SOAP API″
- ″Warning: no string named [rwsu_soapapi_errors] found.″

## Programming the UDDI SOAP API

To use the SOAP API construct a properly formed UDDI message within the body of a SOAP request, and send it using HTTP POST to the URL of the API which the request relates to. The response will be returned within the body of the HTTP reply. Although the samples are written in Java, you may use other programming languages to create your SOAP client, providing you still send requests compliant to the SOAP specification. Valid UDDI requests should conform to the UDDI schema, and be as detailed within the UDDI standard documentation available from:

http://www.uddi.org/

For more information on using the SOAP API, refer to ″The UDDI Registry application programming interface″ section within this InfoCenter.

## SOAP API error handling tips in the UDDI Registry

When using the SOAP API there are three main categories that may cause an error to be returned:-

- An invalid/incorrect request being sent to the SOAP API. eg. Incorrectly formed XML, Badly formed UDDI requests, Non-schema compliant requests.
- Invalid business logic within a SOAP API request. eg. Attempting to delete a business that does not exist.
- Problems occurring while processing a valid request. eg. Server connection to database failure.

In each of these cases, an error will be returned to the client that made the request, which will attempt to explain further what the problem was.

## UDDI Registry application programming interface

The IBM WebSphere UDDI Registry fully supports the Application Programming Interface (API) specification which can be viewed by visiting http://www.uddi.org/pubs/ProgrammersAPI_v2.pdf. Any changes from this specification are documented within the IBM WebSphere UDDI Registry information.

- The Inquiry API
- The Publish API

### Inquiry API for the UDDI Registry

The Inquiry API provides three forms of query that follow broadly used conventions which match the needs of software traditionally used within registries.

- ″Warning: no string named [rwsu_browse_api] found.″
- ″Warning: no string named [rwsu_drilldown] found.″
- ″Warning: no string named [rwsu_invocation_pattern] found.″
- ″Inquiry API functions in the UDDI Registry″

#### Browse pattern for the UDDI Registry

Software that allows people to explore and examine data - especially hierarchical data - requires browse capabilities. The browse pattern characteristically involves starting with some broad information, performing a search, finding general result sets and then selecting more specific information for drill-down.

The UDDI API specifications accommodate the browse pattern by way of the *find_xx* API calls. These calls form the search capabilities provided by the API and are matched with summary return messages that return overview information about the registered information that is associated with the inquiry message type and the search criteria specified in the inquiry.

A typical browse sequence might involve finding whether a particular business you know about has any information registered. This sequence would start with a call to *find_business*, perhaps passing the first few characters of a business name that you already know. This returns a *businessList* result. This result is overview information (keys, names and descriptions) derived from the registered businessEntity information, matching on the name fragment that you provided. If you spot the business you are looking for within this list, you can drill down into the corresponding businessService information, looking for particular technical models (e.g. purchasing, shipping, etc) using the *find_service* API call. Similarly, if you know the technical *fingerprint* (tModel signature) of a particular software interface and want to see if the business you've chosen provides a web service that supports that interface, you can use the find_binding inquiry message.

#### Drilldown pattern for the UDDI Registry

When you have a key for one of the four main data types managed by a UDDI registry, you can use that key to access the full registered details for a specific data instance. The UDDI data types are businessEntity, businessService, bindingTemplate and tModel. You can access the full registered information for any of these structures by passing a relevant key type to one of the *get_xx* API calls.

Continuing the example from the previous section on browsing, one of the data items returned by all of the *find_x* return sets is key information. In the case of the business we were interested in, the businessKey value returned within the contents of a businessList structure can be passed as an argument to *get_businessDetail*. The successful return to this message is a *businessDetail* message containing the full registered information for the entity whose key value was passed. This will be a full *businessEntity* structure.

## Invocation pattern for the UDDI Registry

In order to prepare an application to take advantage of a remote web service that is registered within the UDDI registry by other businesses or entities, you need to prepare that application to use the information found in the registry for the specific service being invoked.

The *bindingTemplate* data obtained from the UDDI registry represents the specific details about an instance of a given interface type, including the location at which a program starts interacting with the service. The calling application or program should cache this information and use it to contact the service at the registered address whenever the calling application needs to communicate with the service instance. In previously popular remote procedure technologies tools have automated the tasks associated with caching (or hard coding) location information. Problems arise however when a remote service is moved without any knowledge on the part of the callers. Moves occur for a variety of reasons, including server upgrades, disaster recovery, and service acquisition and business name changes.

When a call fails using cached information previously obtained from a UDDI Registry, the proper behavior is to query the UDDI Registry for fresh bindingTemplate information. The proper call get_bindingDetail passing the original bindingKey value. If the data returned is different from the cached information, the service invocation should automatically retry the invocation using the fresh information. If the result of this retry is successful, the new information should replace the cached information.

By using this pattern with web services, a business using a UDDI Registry can automate the recovery of a large number of partners without undue communication and coordination costs. For example, if a business has activated a disaster recovery site, most of the calls from partners will fail when they try to invoke services at the failed site. By updating the UDDI information with the new address for the service, partners who use the invocation pattern will automatically locate the new service information and recover without further administrative action.

## Inquiry API functions in the UDDI Registry

These messages represent inquiries that can be made of the UDDI Registry. These messages all behave synchronously.

The queries available are:

**find_binding**
: Used to locate specific bindings within a registered businessService. Returns a bindingDetail message.

**find_business**
: Used to locate information about one or more businesses. Returns a businessList message.

**find_relatedBusinesses**
: Used to locate information about businessEntity registrations that are related to a specific business entity whose key is passed in the inquiry. The Related Businesses feature is used to manage registration of business units and subsequently relate them based on organizational hierarchies or business partner relationships. Returns a relatedBusinessList message.

**find_service**
: Used to locate specific services within a registered businessEntity. Returns a serviceList message.

**find_tModel**
: Used to locate one or more tModel information structures. Returns a tModelList structure.

**get_bindingDetail**
: Used to get full bindingTemplate information suitable for making one or more service requests. Returns a bindingDetail message.

**get_businessDetail**
: Used to get the full businessEntity information for one or more businesses or organizations. Returns a businessDetail message.

**get_serviceDetail**
: Used to get full details for a given set of registered businessService data. Returns a serviceDetail message.

**get_tModelDetail**
: Used to get full details for a given set of registered tModel data. Returns a tModelDetail message.

**Accessible query values in the UDDI Registry:** A list of the accessible queries for the UDDI Registry is given here.

Accessible queries within the UDDI Registry are:

**find_binding**
: used to locate specific bindings within a registered businessService. Returns a bindingDetail message that contains zero or more bindingTemplate structures matching the criteria specified in the arguement list.

**find_business**
: used to locate information about one or more businesses. Returns a businessList message that matches the conditions specified in the arguments.

**find_relatedBusinesses**
: used to locate information about businessEntity registrations that are related to a specific business entity whose key is passed in the inquiry. The Related Businesses feature is used to manage registration of business units and subsequently relate them based on organizational hierarchies or business partner relationships. Returns a relatedBusinessList message containing results that match the conditions specified in the arguments.

**find_Service**
: used to locate specific services within a registered businessEntity. Returns a serviceList message that matches the conditions specified in the arguments.

**find_tModel**
: used to locate a list of tModels that match a set of specified criteria. The response will be a list of abbreviated information about registered tModel data that amtches the criteria specified.The result will be returned in a tModelList message.

**get_bindingDetail**
: used to requesting the run-time bindingTemplate information for the purpose of invoking a registered business API. Returns a bindingDetail message.

**get_businessDetail**
: used to return complete businessEntity information for one or more specified businessEntity registrations matching on the businessKey values specified.

**get_businessDetailExt**
: used to return extended businessEntity information for one or more specified businessEntity registrations. This message returns exactly the same information as the get_businessDetail message, but may contain additional attributes if the source is an external registry that is compatible with the API specification.

**get_serviceDetail**
: used to request full information about a known businessService structure. Returns a serviceDetail message.

**get_tModelDetail**
: get_tModelDetail

For full details of the syntax of the above queries, please refer to the API Specification at http://www.uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf

## Publish API for the UDDI Registry

The messages in this section represent commands that are used to publish and update information contained in a UDDI registry. The messages defined in this section all behave synchronously.

The Publishing API calls defined that UDDI operators support are:

**add_publisherAssertions**
: this call causes one or more publisherAssertion to be added to an individual publisher's assertion collection.

**delete_binding**
: causes one or more instances of bindingTemplate data to be deleted from the UDDI registry.

**delete_business**
: used to remove one or more business registrations and all direct contents from a UDDI registry.

**delete_publisherAssertions**
: causes one or more publisherAssertion elements to be removed from a publisher's assertion collection.

**delete_service**
: is used to remove one or more businessService elements from the UDDI registry and from its containing businessEntity parent.

**delete_tModel**
: is used to logically delete one or more tModel structures. Logical deletion hides the deleted tModels from find_tModel result sets but does not physically delete it.

**discard_authToken**
: is used to inform an Operator Site that the authentication token is to be discarded, effectively ending the session. Subsequent calls that use the

same authToken will be rejected. This message is optional for Operator Sites that do not manage session state or that do not support the get_authToken message.

**get_assertionStatusReport**
: this call provides administrative support for determining the status of current and outstanding publisher assertions that involve any of the business registrations managed by the individual publisher account. Using this message, a publisher can see the status of assertions that they have made, as well as see assertions that others have made that involve businessEntity structures controlled by the calling publisher account.

**get_authToken**
: the call used to obtain an authentication token.Authentication tokens are opaque values that are required for all other publisher API calls. This message is not required for Operator Sites that have an external mechanism defined for users to get an authentication token. This API is provided for implementations that do not have some other method of obtaining an authentication token or certificate, or that choose to use user ID and password based authentication.

**get_publisherAssertions**
: this is used to obtain the full set of publisher assertions that are associated with an individual publisher account. Publisher assertions are used to control publicly visible business relationships.

**get_registeredInfo**
: this call is used to get an abbreviated list of all businessEntity and tModel data that are controlled by the individual associated with the credentials passed.

**save_binding**
: is used to save or update a complete bindingTemplate element. this message can be used to add or update one or more bindingTemplate elements as well as the container/contained relationship that each bindingTemplate has with one or more existing businessService elements.

**save_business**
: this is used to save or update information about a complete businessEntity element. This API has the broadest scope of all the save_xx API calls in the publisher API, and can be used to make sweeping changes to the published information for one or more businessEntity elements controlled by an individual.

**save_service**
: the call used to add or update one or more businessService elements exposed by a specified businessEntity.

**save_tModel**
: this call adds or updates one or more registered tModel elements.

**set_publisherAssertions**
: this call is used to manage all of the tracked relationship assertions associated with an individual publisher account.

For full details of the syntax of the above queries, please refer to the API Specification at http://www.uddi.org/pubs/ProgrammersAPI_v2.pdf.

## UDDI EJB Interface for the UDDI Registry

This section describes how to use the EJB application programming interface (API) of the IBM WebSphere UDDI Registry component to publish, find and delete UDDI entries.

The necessary client classes are contained in the *uddiejbclient.jar* file in the ejb subdirectory of the UDDIReg directory under the WebSphere appserver directory tree.

The javadoc for the EJB API is contained in the javadoc directory tree under the ejb subdirectory of the UDDIReg directory under the WebSphere appserver directory tree.

The EJB API is contained in two stateless session beans, one for the Inquiry API (com.ibm.uddi.ejb.InquiryBean) and one for the Publish API (com.ibm.uddi.ejb.PublishBean), whose public methods form an EJB interface for the UDDI Registry. All the public methods on the InquiryBean correspond to UDDI Inquiry API functions, and all the public methods on the PublishBean correspond to UDDI Publish API functions. (Not all UDDI API functions are implemented, e.g. get_authToken, discard_authToken, get_businessDetailExt, etc.) For version 1 of the UDDI registry, the EJB component supports only UDDI v2.0.

The two EJBs use container-managed transactions. The transaction attribute for the methods of the InquiryBean is NotSupported, and for the methods of the PublishBean it is Required. You should not change the transaction attributes as this could result in undesirable behavior.

Within each interface there are groups of overloaded methods that correspond to the operations in the UDDI 2.0 specification. There is a separate method for each major variation in function. For example, the single UDDI 2.0 operation find_business is represented by 10 variations of findBusiness methods, with different variations for finding by name, finding by categoryBag etc.

The arguments for the EJB interface methods are java objects in the package com.ibm.uddi.datatypes. Roughly speaking, there is a one-one correspondence between classes in this package and elements of the UDDI v2.0 XML schema. Exceptions to this are, for example, where UDDI XML elements can be represented by a single String. (See Package com.ibm.uddi.datatypes below for more information.)

**Enabling an EJB Client**

This section is written on the assumption that WebSphere Application Server V5.0, a supported database and the IBM WebSphere UDDI Registry have already been installed.

**Classpaths**

In order for EJB clients to work correctly, the following jar files and folders must be added to the user's CLASSPATH:

**For Windows**

```
<WebSphere-install-dir>\lib\j2ee.jar
<WebSphere-install-dir>\lib\naming.jar
```

<WebSphere-install-dir>\lib\namingclient.jar
<WebSphere-install-dir>\lib\ecutils.jar
<WebSphere-install-dir>\lib\sas.jar
<WebSphere-install-dir>\properties

**For Unix Platforms**

<WebSphere-install-dir>/lib/j2ee.jar
<WebSphere-install-dir>/lib/naming.jar
<WebSphere-install-dir>/lib/namingclient.jar
<WebSphere-install-dir>/lib/ecutils.jar
<WebSphere-install-dir>/lib/sas.jar
<WebSphere-install-dir>/properties

In addition to these jars, there is also the jar file that contains all of the UDDI specific API for the EJB interface, which can be found at:

**For Windows**

<DeploymentManager-install-dir>\UDDIReg\ejb\uddiejbclient.jar

where <DeploymentManager-install-dir> is the install location for WebSphere Application Server for Network Deployment, which by default is C:\Progra~1\WebSphere\DeploymentManager.

**For Unix Platforms**

<DeploymentManager-install-dir>/UDDIReg/ejb/uddiejbclient.jar

where <DeploymentManager-install-dir> is the install location for WebSphere Application Server for Network Deployment, which by default is */opt/WebSphere/DeploymentManager* for Linux/Solaris systems or */usr/WebSphere/DeploymentManager* for AIX systems.

**The Path**

Please ensure that your PATH statement starts with <WebSphere-install-dir>\java\bin

**Creating an EJB Client**

If you want to read about creating EJB Clients in more detail, then please read the ″Sun Microsystems Enterprise JavaBeans$^{TM}$ Specification Version 2.0″

**Finding the EJB Reference**

An EJB Client can be a standalone Java application, an Applet, Servlet or a JSP. This document only covers writing a standalone Java application. In order to invoke an EJB that has been deployed into WebSphere on the server side, the Client must do two things: find the EJB on the server, and then create a Cient side reference to that EJB. Once this Client side reference has been created, then the

Client can invoke methods upon the EJB as if it was a local object. Clients cannot reference, or invoke, and EJB directly. Any calls made to the EJB must be made through the interfaces that the EJB provides. The interface that is used to create a local reference to the EJB is called the *home interface*. When an EJB is deployed in WebSphere, this home interface is made available to Clients by means of a searchable namespace. This means that a Client can look up an address on the namespace. If there is a home interface at that address, and it is the home interface to the EJB that they were looking for, then the Client can create a local instance of that home interface, and then, from that, a local reference to the EJB can be created.

**What code is needed in the Client?**

The following code fragment illustrates how to Find and Create a local instance of the Inquiry EJB only. The same will need to be done to Find and Create a local copy of the Publish EJB.

```
private com.ibm.uddi.ejb.Inquiry inquiry = null;   // This private variable,
             // "inquiry" is going to be the local reference to the EJB in
             // WebSphere declaring it outside the scope of a method means
             // that this same reference can be used throughout the client,
             // without having to query the namespace again.
public void homeLookup()
{
 // These variables simply determine the address of the JNDI namespace, and
   // the address of the home interface within that namespace.

 // String naming_factory = "com.ibm.ejs.ns.jndi.CNInitialContextFactory";
      //WAS 4.0.2 Naming Factory
 String naming_factory = "com.ibm.websphere.naming.WsnInitialContextFactory";
      //WAS 5.0 Naming Factory

 String namespace_address = "iiop://localhost:2809/";
      // The address of the namespace
 String home_address = "com/ibm/uddi/ejb/InquiryHome";
      // The address of the home interface within the JNDI namespace

 java.util.Hashtable environment = new java.util.Hashtable();
 environment.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY, naming_factory);
 environment.put(javax.naming.Context.PROVIDER_URL, namespace_address);

 try
 {
  javax.naming.InitialContext ic = new javax.naming.InitialContext(environment);
        // Create a context using the details above to connect to the namespace

      Object o = ic.lookup(home_address);
          // Do a lookup to see if there is an ejb_home at the address given above

      // Now create a valid home instance for the EJB type we want to create
  com.ibm.uddi.ejb.InquiryHome home = (com.ibm.uddi.ejb.InquiryHome)
          (javax.rmi.PortableRemoteObject.narrow(
            o, com.ibm.uddi.ejb.InquiryHome.class));

      inquiry = home.create();
           // Now create a local reference of the EJB, by using the home.create()
           // method.  Any business method that is intended for the EJB in
            // Websphere must me invoked against this inquiry object.
 }
 catch (javax.naming.NamingException ne) {ne.printStackTrace();}
       // This is thrown if there was a problem connecting to the namespace,
       // or finding the home_address in the namespace
 catch (java.rmi.RemoteException re) {re.printStackTrace();}
       // This usually indicates some sort of system failure, either
       // WebSphere is not running, or there is a communications problem
 catch (javax.ejb.CreateException ce) {ce.printStackTrace();}
       // This is thrown if the EJB reference cannot be created from
       // the home instance.
}
```

### Writing Client code to use the EJB API

Once the reference to the EJB has been created (the Inquiry Object, in the above code), then the reference can be treated like any other Java Object. This is an example method using the UDDI EJB API - the only important point to remember is that, although the Inquiry Object has been created as a local reference, it is still referring to a remote EJB Object in a different server, possibly even in a different country. This means that at the very least a javax.rmi.RemoteException must be caught on each method call that is made to the EJB.

```
public void findBusiness()
{
 System.out.println("Find Business:");
         NameList names = new NameList();
         names.add(new Name("IBM Corporation"));
  // Create the list of names to find in the UDDI Registry, here just
      // one is used, "IBM Corporation"
         try
         {
             BusinessList list = inquiry.findBusiness(names);
                // This is the call to the inquiry EJB that searches through
                // the UDDI Registry

  // Now display the amount of business found, and for each one, get the
      // BusinessKey, the BusinessName and the amount of Services that
      // Business has
             System.out.println("There are "+list.getBusinessInfos().size()+"
                   matching Businesses in this registry");
             for (int i=0;i<list getBusinessInfos().size();i++)
       {
                  BusinessInfo business = list.getBusinessInfos().get(i);
                  System.out.println("\nBusinessKey = "+business.getBusinessKey());
                  System.out.println("BusinessName =
                      "+business.getNames().get(0).getNameString());
                  System.out.println("This Business Has
                      "+business.getServiceInfos().size()+" Services\n");
             }
         }

 // This is a UDDI specific exception, and will be thrown if for example
   // an invalid name was used as the search criteria
 catch (com.ibm.uddi.datatypes.DispositionReportException e)
      {this.handleDispositionReportException(e);}

         catch (java.rmi.RemoteException re) {re.printStackTrace();}
          // This is the RemoteException that is thrown if there has been
          // a system failure or a connection problem.
}
```

### What new code is needed on the Client?

Just as each EJB has an interface listed on the JNDI namespace, the javax.transaction.UserTransaction class also has an interface listed. This means that the same method used to get a local instance of an EJB can be applied to get a local instance of the UserTransaction class. Again, this code can be used to find the UserTransaction reference on the namespace, in addition to the code required to find the Inquiry EJB and the Publish EJB, or, alternatively, there is a slightly more elegant method used in the TransactionEJBClientSample.java.

```
public void txLookup()
{
 private javax.transaction.UserTransaction tx = null;
    // This is the private variable that will be used to hold the
    // UserTransaction Object declaring it outside the scope of a method
    // means that this same reference can be used throughout the client,
    // without having to query the namespace again.


 // These variables simply determine the address of the JNDI namespace, and
   // the address of the home interface within that namespace.

 // String naming_factory = "com.ibm.ejs.ns.jndi.CNInitialContextFactory";
     // WAS 4.0.2 Naming Factory
 String naming_factory = "com.ibm.websphere.naming.WsnInitialContextFactory";
     // WAS 5.0 Naming Factory

 String namespace_address = "iiop://localhost:2809/";
      //The address of the namespace
 String transaction_address = "jta/usertransaction";
      //The address of the UserTransaction interface within the JNDI namespace

 java.util.Hashtable environment = new java.util.Hashtable();
 environment.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY, naming_factory);
 environment.put(javax.naming.Context.PROVIDER_URL, namespace_address);


 try
 {
  javax.naming.InitialContext ic = new javax.naming.InitialContext(environment);
        // Create a context using the details above to connect to the namespace
      Object remote_object = ic.lookup(transaction_address);
            // Do a lookup to see if there is a UserTransaction Object at the
            // address specified above
  tx = (javax.transaction.UserTransaction)remote_object;
         // Convert the remote object found into a UserTransaction Object,
         // and assign to the private variable
 }
 catch (javax.naming.NamingException ne) {ne.printStackTrace();}
      // This is thrown if there was a problem connecting to the namespace,
      // or finding the transaction_address in the namespace
}
```

## Writing Client code to use the EJB API with a Client transaction

To perform an Inquiry, a Publish or a Delete upon the IBM WebSphere UDDI Registry with client side transactional support requires very little additional code compared to doing the same operations without client side transactional support. Using the same code that is listed above (in ″Writing Client Code to use the EJB API″), this example illustrates how easy client side transactions are to implement.

The additional lines of code needed are in bold type. This code also assumes that there is a variable called *tx* that has been declared at the class scope.

```
public void findBusiness()
{
//Just as there are UDDI and RMI specific exceptions thrown,
// 5 more exceptions need to be caught.
try
{
tx.begin(); //This begins the transaction context
System.out.println("Find Business:");
NameList names = new NameList();
names.add(new Name("IBM Corporation"));
  // Create the list of names to find in the UDDI Registry,
  // here just one is used, "IBM Corporation"
try
{
BusinessList list = inquiry.findBusiness(names);
 // This is the call to the inquiry EJB that searches through the UDDI Registry

// Now display the amount of business found, and for each one, get the
// BusinessKey, the BusinessName and the amount of Services that Business has
System.out.println("There are "+list.getBusinessInfos().size()+"
   matching Businesses in this registry");
for (int i=0;i<list.getBusinessInfos().size();i++)
{
BusinessInfo business = list.getBusinessInfos().get(i);
System.out.println("\nBusinessKey = "+business.getBusinessKey());
System.out.println("BusinessName = "+business.getNames().get(0).getNameString());
System.out.println("This Business Has "+business.getServiceInfos().size()+"
  Services\n");
}
}
// This is a UDDI specific exception, and will be thrown if for example an
// invalid name was used as the search criteria
catch (com.ibm.uddi.datatypes.DispositionReportException e)
   {this.handleDispositionReportException(e);}
catch (java.rmi.RemoteException re) {re.printStackTrace();}
  // This is the RemoteException that is thrown if there has been a
  // system failure or a connection problem.

tx.commit(); //This ends the transaction context
}
catch (javax.transaction.NotSupportedException nse)
   {nse.printStackTrace();}
catch (javax.transaction.RollbackException rbe) {rbe.printStackTrace();}
catch (javax.transaction.SystemException se) {se.printStackTrace();}
catch (javax.transaction.HeuristicMixedException hme)
   {hme.printStackTrace();}
catch (javax.transaction.HeuristicRollbackException hrbe)
   {hrbe.printStackTrace();}
}
```

- ″Datatypes package in the UDDI Registry″
- ″EJB Interface Methods in the UDDI Registry″

## Datatypes package in the UDDI Registry

Below is a table listing the classes in the com.ibm.uddi.datatypes package, the elements in the UDDI v2.0 XML schema, and the correspondence between the two.

| com.ibm.uddi.datatypes Class | Corresponding UDDIv2.0 XML Schema Element | Notes on DatatypeClass |
|---|---|---|
| AccessPoint | accessPoint | |
| Address | address | |
| String | addressLine | |
| AdressLineList | | Encapsulates a Vector of addressLine Strings |
| AddressList | | Encapsulates a Vector of Address objects |
| AssertionStatusItem | assertionStatusItem | |
| AssertionStatusItemList | | Encapsulates a Vector of AssertionStatusItem objects |
| AssertionStatusReport | assertionStatusReport (response message) | |
| String | authInfo | |
| AuthToken | | Object containing authInfo String and operator name |
| String | bindingKey | |

| | | |
|---|---|---|
| BindingDetail | bindingDetail (response message) | |
| BindingTemplate | bindingTemplate | |
| BindingTemplateList | bindingTemplates | Encapsulates a Vector of Bindingtemplate objects |
| BusinessDetail | businessDetail (reponse message) | |
| BusinessDetailExt | businessDetailExt (Response message) | ** |
| BusinessEntity | businessEntity | |
| BusinessEntityExt | businessEntityExt | ** |
| BusinessEntityExtList | | Encapsulates a Vector of BusinessEntityExt objects ** |
| BusinessEntityList | | Encapsulates a Vector of BusinessEntity objects |
| BusinessInfo | businessInfo | |
| BusinessInfoList | businessInfo | Encapsulates a Vector of businessInfo objects |
| String | businessKey | |
| BusinessList | businessList (reponse message) | |
| BusinessService | businessService | |
| BusinessServiceList | businessServices | Encapsulates a Vector of BusinessService objects |
| CategoryBag | categoryBag | |
| String | completionStatus | |
| Contact | contact | |
| ContactList | contacts | Encapsulates a Vector of Contact objects |
| Description | description | |
| DescriptionList | | Encapsulates a Vector of Description objects |
| DiscoveryUrl | discoveryURL | |
| DiscoveryUrlList | discoveryURLs | Encapsulates a Vector of DiscoveryURL objects |
| DispositionReport | dispositionReport | |
| DispositionreportException | | Exception thrown by EJB interface functions when an error occurs |
| Email | email | |
| EmailList | | Encapsulates a Vector of Email objects |
| EndPoint | | Used as baseclass for AccessPoint and HostingRedirector providing mutual exclusivity |
| ErrInfo | errInfo | |
| | findQualifier | |
| FindQualifier | findQualifiers | |
| String | fromKey | |
| HostingRedirector | hostingRedirector | |
| IdentifierBag | identifierbag | |
| InquiryOptions | | Encapsulates a FindQualifiers object and a maxrows field. Used in find_* API calls to specify search options |
| InstanceDetails | instanceDetails | |

| | | | |
|---|---|---|---|
| | String | instanceParms | |
| | String | keyValue | |
| KeyedReference | | keyedReference | |
| keysOwned | | keysOwned | |
| LanguageString | | | Abstract class, extended by some of the datatypes, which represents a string that can optionally be tagged with xml:lang. |
| Name | | name | |
| NameList | | | Encapsulates a Vector of Name objects |
| OverviewDoc | | overviewDoc | |
| | String | overviewURL | |
| | String | personName | |
| Phone | | phone | |
| PhoneList | | | Encapsulates a Vector of Phone objects |
| PublisherAssertion | | publisherAssertion | |
| PublisherAssertionList | | | Encapsulates a Vector of Publisher Assertion objects |
| PublisherAssertions | | publisherAssertions (response message) | |
| RegisteredInfo | | registeredInfo (response message) | |
| | | relatedBusinessInfo | not used |
| | | relatedBusinessInfos | not used |
| RelatesBusinessesList | | relatedBusinessesList | |
| RelatedBusinessInfo | | relatedBusinessInfo | |
| RelatedBusinessInfos | | relatedBusinessInfos | |
| Result | | result | |
| ResultList | | | Encapsulates a Vector of Result objects |
| ServiceDetail | | serviceDetail (response message) | |
| ServiceInfo | | serviceInfo | |
| ServiceInfoList | | serviceInfos | Encapsulates a Vector of serviceInfo objects |
| | String | serviceKey | |
| ServiceList | | serviceList (reponse message) | |
| | | sharedRelationships | not used |
| SharedRelationships | | sharedRelationships | |
| Tmodel | | tModel | |
| TModelBag | | tModelBag | |
| TModelDetail | | tModelDetail (response message) | |
| TModelInfo | | tModelInfo | |
| TModelInfoList | | tModelInfos | Encapsulates a Vector of TModelInfo objects |
| TModelInstanceInfo | | tModelInstanceInfo | |
| TModelInstanceInfoList | | tModelInstanceDetails | Encapsulates a Vector of TModelInstanceInfo objects |
| | String | tModelKey | |
| TModelList | | tModelList (response message) | |
| TModels | | | Encapsulates a Vector of TModel objects |

|  |  |  |  |
|---|---|---|---|
|  | String | toKey |  |
|  | String | uploadRegister |  |
| UploadRegisterList |  |  | Encapsulates a Vector of uploadRegister strings |

** Used in UDDI API functions relating to BusinessDetailExtension. These UDDI API functions are not implemented in Version 1 of the IBM WebSphere UDDI Registry.

In general, a datatype called DatatypeList contains a vector of *Datatype* objects. Often these correspond to XML schema elements with plural names. (For example the datatype *Contact* corresponds to XML element *contact*, and *ContactList* corresponds to *contacts*.) Where there is no *plural* XML schema element for a particular *Datatype*, often there is still a *DatatypeList* where it is useful to have one, e.g. *AddressList*.

The exceptions to this naming convention occur when there is an existing XML schema element ending in List. The exceptions are: TModelList, ServiceList, BusinessList. In these cases, the corresponding datatypes are given the same names as the XML schema elements, and the datatypes that would have had these names are called: TModels, BusinessServiceList, BusinessEntityList.

## EJB Interface Methods in the UDDI Registry

**Inquiry**

```
findBinding
findBusiness
findRelatedBusinesses
findService
findTModel
getBindingDetail
getBusinessDetail
getServiceDetail
getTModelDetail
```

**Publish**

```
addPublisherAssertions
deleteBinding
deleteBusiness
deletePublisherAssertions
deleteService
deleteTModel
getAssertionStatusReport
getRegisteredInfo
getPublisherASsertions
saveBinding
saveBusiness
saveService
saveTModel
setPublisherAssertions
```

Each method is overloaded and can take various combinations of arguments. The Javadoc contains detailed information about each method.

Note that get_authToken and discard_authtoken are not implemented, as WebSphere security is used instead.

## UDDI Troubleshooting Tips

When the IBM WebSphere UDDI Registry is running, it might issue messages to report events or errors. You can use these messages, described in Messages as your first aid to problem determination. If you need more details about the causes of a problem, you can turn on tracing for UDDI, as described in Turning on UDDI trace.

- ″Turning on UDDI trace″

Below are a few of the common causes of errors that might be found and their suggested solutions.

- If you set up the UDDI Registry application with a JDBC driver and datasource that reference Cloudscape, but set the persister property in *uddi.properties* to specify DB2, **or vice versa**, then some unexpected behavior will result, such as a fatal error on deleting an entity. If this happens, you should check that the above details are not in conflict. This only applies to a UDDI Registry installation on a single appserver.
- If you get a message ″The application failed to initialize″ when trying to access the UDDI User console and you are using DB2 as the persistence store for the UDDI Registry, a likely cause of the problem is that you specified the wrong userid and/or password when you ran the script to install the UDDI Registry application. If this occurs rerun the script ensuring you use the correct userid and password.
- You might find that, after uninstalling and reinstalling the UDDI Registry, you get errors from the UDDI User Console of the form:

  ″Error 500: JSPG0059E: Unable to compile class for JSP″.

  If this occurs, then you should clear out the temp directory of the WebSphere AppServer.
- When running one of the UDDI setup scripts setupuddi.jacl or removeuddi.jacl, if you get an error such as:

  WASX7017E: Exception received while running file ″setupuddi.jacl″; exception information: com.ibm.bsf.BSFException: error while eval′ing Jacl expression: java.util.MissingResourceException: Can′t find resource for bundle java.util.PropertyResourceBundle, key ErrMsgIncorrectNumArgs

  then please ensure that the file setupuddimessages.jafr is located in the *lib* subdirectory of the WebSphere deployment manager or application server under which you are running the script.
- When running the DB2 Setup Wizard, if you get an error stating ″Invalid userid and password″, then if could be caused by any of the following situations:
  - You have supplied an invalid userid or password - re-enter with a valid userid and password.
  - The supplied userid does not have the necessary privileges - retry with a userid that has appropriate privileges.
  - DB2 is stopped when you run the Wizard - start DB2 and retry the Wizard.
  - The UDDI20 database already exists and has been removed previously and, as such, is not catalogued. The DB2 Wizard does not recognize this situation and gives the error. You now have two options.
    1. If you wish to use the existing database then you will need to catalogue it and there is no need to rerun the Wizard.
    2. If you wish to create a new database you will need to recatalogue the database and re-run the DB2 Wizard and choose the option to overwrite the database. (Any existing data **WILL** be lost.)

## Turning on UDDI trace

Before you begin

You can enable UDDI-specific trace in the same way as you enable other tracing in the WebSphere Application Server.

The following is a list of trace strings that may be used:

- com.ibm.uddi.api
- com.ibm.uddi.config
- com.ibm.uddi.datatypes
- com.ibm.uddi.dom
- com.ibm.uddi.ejb
- com.ibm.uddi.exception
- com.ibm.uddi.exceptions
- com.ibm.uddi.gui
- com.ibm.uddi.gui.inquire
- com.ibm.uddi.gui.publish
- com.ibm.uddi.persistence
- com.ibm.uddi.persistence.jdbc
- com.ibm.uddi.persistence.jdbc.cloudscape
- com.ibm.uddi.persistence.jdbc.db2
- com.ibm.uddi.ras
- com.ibm.uddi.security
- com.ibm.uddi.soap
- com.ibm.uddi.uuid
- com.ibm.uddi.validation
- com.ibm.uddi.xml

For example, to trace the UDDI User Console you would specify:

```
'com.ibm.uddi.gui=all=enabled'
```

This would enable all types of trace for the gui. Please refer to ″Enabling trace″ elsewhere in the WebSphere Application Server V5.0 InfoCenter for more information about using the administrator console to enable/disable trace.

## Messages

When the IBM WebSphere UDDI Registry is running, it might issue messages to report events or errors. The messages are in the form `UD<i>xxnnnns</i>` where:

| | |
|---|---|
| **xx** | is a two character descriptor identifying which component is involved |
| **nnnn** | give the error code being issued |
| **s** | is either I (Information) or E (Error) |

The prefix *UDxxnnnns:* is followed by text which describes the event or error. For some messages, the first word of the text is one of the form (MSN=SSSS). The SSSS provides a message sequence number (or MSN), which identifies the unique circumstance in which the message was issued, and is of use where the same message can be issued in more than one circumstance.

To help you diagnose problems and minimize the need to enable trace in any of the above components, view the messages table. You can view the messages by prefix or component, whichever is easiest for you to find in the table. All messages are documented with user/system action and explanation.

The text for the UDDI messages is contained in a file *uddiresourcebundles.jar* which is placed, by the installation process, into the \lib subdirectory (Windows) of the WebSphere application server into which the UDDI Registry was installed. If you will be running a console or log analyzer from another process; for example, to analyze the activity log, then you must place a copy of *uddiresourcebundles.jar* into a directory which is within the classpath of that process. Otherwise, the message lookup for the UDDI messages will fail.

**UDDI Components Message Prefix Table**

| **click on individual links for message documentation for the component** | |
|---|---|
| `<a href="#rwsu_udai">UDAI</a>` | `<a href="#rwsu_udai">API</a>` |
| `<a href="#rwsu_udcf">UDCF</a>` | `<a href="#rwsu_udcf">Configuration</a>` |
| `<a href="#rwsu_udda">UDDA</a>` | `<a href="#rwsu_udda">Datatypes</a>` |
| `<a href="#rwsu_uddm">UDDM</a>` | `<a href="#rwsu_uddm">DOM</a>` |
| `<a href="#rwsu_udej">UDEJ</a>` | `<a href="#rwsu_udej">EJB Interface</a>` |
| `<a href="#rwsu_udex">UDEX</a>` | `<a href="#rwsu_udex">Exceptions</a>` |
| `<a href="#rwsu_udin">UDIN</a>` | `<a href="#rwsu_udin">Installation</a>` |
| `<a href="#rwsu_udlc">UDLC</a>` | `<a href="#rwsu_udlc">Local API</a>` |
| `<a href="#rwsu_udpr">UDPR</a>` | `<a href="#rwsu_udpr">Persistence</a>` |
| `<a href="#rwsu_udrs">UDRS</a>` | `<a href="#rwsu_udrs">Logging</a>` |
| `<a href="#rwsu_udsc">UDSC</a>` | `<a href="#rwsu_udsc">Security</a>` |
| `<a href="#rwsu_udsp">UDSP</a>` | `<a href="#rwsu_udsp">SOAP Interface</a>` |
| `<a href="#rwsu_uduc">UDUC</a>` | `<a href="#rwsu_uduc">User Console</a>` |
| `<a href="#rwsu_uduu">UDUU</a>` | `<a href="#rwsu_uduu">UUID</a>` |

## UDAI (Web Services UDDI) messages

There are no messages issued by this component.

## UDCF (Web Services UDDI) messages

**UDCF0001E: Exception occurred while getting int value of configuration property ″<property>″, exception: ″<exception>″**

**Explanation:** This message is issued when an attempt to read the value of a configuration property from the uddi.properties file and convert it to integer has failed with the indicated exception.

**User Response:** Check that the uddi.properties file contains a value for the indicated configuration property, and that the value is valid. Check also that the indicated configuration property is a legal property. Refer to the InfoCenter for further information about global configuration properties and the uddi.properties file.

**UDCF0002E: Exception occurred while getting long value of configuration property ″<property>″, exception: ″<exception>″**

**Explanation:** This message is issued when an attempt to read the value of a configuration property from the uddi.properties file and convert it to long has failed with the indicated exception.

**User Response:** Check that the uddi.properties file contains a value for the indicated configuration property, and that the value is valid. Check also that the indicated configuration property is a legal property. Refer to the InfoCenter for further information about global configuration properties and the uddi.properties file.

**UDCF0003E: Exception occurred while getting boolean value of configuration property ″<property>″, exception: ″<exception>″**

**Explanation:** This message is issued when an attempt to read the value of a configuration property from the uddi.properties file and convert it to boolean has failed with the indicated exception

**User Response:** Check that the uddi.properties file contains a value for the indicated configuration property, and that the value is valid. Check also that the indicated configuration property is a legal property. Refer to the InfoCenter for further information about global configuration properties and the uddi.properties file.

**UDCF0004E: Failed to load UDDI global properties file.**

**Explanation:** This message is issued when the UDDI global configuration properties file, uddi.properties, cannot be loaded. Default values for the global configuration properties will be set, but these defaults may not be adequate for many of the properties, so you should investigate and resolve this problem.

**User Response:** Check that the uddi.properties file exists and is in the correct directory. Refer to the InfoCenter for further information about global configuration properties and the uddi.properties file.

**UDCF0005E: Exception occurred while loading UDDI global configuration properties, exception: ″<exception>″**

**Explanation:** This message is issued when an attempt to load the UDDI global configuration properties from the uddi.properties has failed with the indicated exception. Default values for the global configuration properties will be set, but these defaults may not be adequate for many of the properties, so you should investigate and resolve this problem.

**User Response:** Check that the uddi.properties file exists and contains valid values for each of the configuration properties. Refer to the InfoCenter for further information about global configuration properties and the uddi.properties file.

## UDDA (Web Services UDDI) messages

There are no messages issued by this component.

## UDDM (Web Services UDDI) messages

There are no messages issued by this component.

## UDEJ (Web Services UDDI) messages

There are no messages issued by this component.

## UDEX (Web Services UDDI) messages

There are no messages issued by this component.

## UDIN (Web Services UDDI) messages

**UDIN0001I: Assuming hard coded defaults.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None..

**UDIN0002I: Cloudscape classpath is clpath. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0003I: Looking for childtype childname under parenttype parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0004I: Looking for childtype childname under parenttype parentname and parenttype2 parentname2. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0005I: Conflict found with existing childtype childname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0006I: Not creating requested childtype. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0007I: Seeking parenttype with requested id of parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0008I: Seeking parenttype with requested id of parentname under parenttype2 parentname2. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0009I: Attempting to create childtype under parenttype of parentID. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0010I: Create command that will be issued is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0011I: childtype childId was successfully created. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0012I: Looking for builtin_rra.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0013I: List for J2CResourceAdapter returned N members. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0014I: Hunting J2CResourceAdapter associated with Node nodename. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0015I: Using rraID as builtin_rra. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0016I: Using provider class of implclass with a classpath of clpath. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0017I: Installing to server servername, node nodename using database type of dbtype. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0018I: Attempting to create UDDI JDBCProvider.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0019I: Attempting to create UDDI Datasource.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0020I: Application Manager appmgr found. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0021I: Attempting to install UDDI Registry application.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0022I: Checking for installed UDDI Registry application of name appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0023W: Application of name appname is not present. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0024I: ApplicationManager not running, so application will not need to be stopped.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0025I: Stopping application of name appname. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0026W: stopApplication command for application appname caught exception Exc. Application might not have been running on this server. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0027I: Application appname stopped successfully. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0028I: Removing application appname. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0029I: Application appname removed successfully. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0030I: Adding resource bundles to repository.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0031I: Adding Cloudscape user functions to repository.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0032I: UDDI configuration properties file already exists. Only the persister and getServletURLPrefix properties will be overwritten.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0033I: Editing UDDI configuration properties file propsfile. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0034I: Url prefix found. Updating it to discoveryURL. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0035I: Persister property found. Updating it to dbtype. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0036I: Adding UDDI configuration properties file to repository for cell cellname under target node and server. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0037I: ws.ext.dir processing starting.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0038I: serverID is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0039I: JVM is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0040I: Out of N properties we located M matches at positions poslist. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0041I: Building new ws.ext.dirs properties.**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0042I: SYSPROP is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0043I: ws.ext.dir has been set with new sysprop. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0044I: ws.ext.dir update skipped, required changes already present.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0045I: ws.ext.dir processing step complete.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0046I: Cleaning up temporary version of properties file temppropsfile. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0047I: Issuing nodeSync.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0048I: UDDI Registry successfully installed. Please restart server servername to activate configuration changes. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0049I: Application Manager appmgr found. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0050I: Server is not running, so will not need to be stopped.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0051I: Stopping server servername under node nodename. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0052I: Server servername stopped successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0053I: Restarting application server**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0054I: Application server servername restarted successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0055I: Please ignore any errors concerning the serverStartupSyncEnabled attribute.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0101I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0102I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0103I: Changes were not saved on this call.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0104I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0105I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0106I: Attempting to save ws.ext.dir changes.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0107I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0108I: Attempting final save of new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN0109I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script setupuddi.jacl.

**User Response:** None.

**UDIN1001I: Application Manager appmgr found. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1002I: Server is not running, so will not need to be stopped.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1003I: Stopping server servername under node nodename. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1004I: Server servername stopped successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1005I: Resource bundles file will be removed from repository if present.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl .

**User Response:** None.

**UDIN1006I: Removing resource bundles from repository.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1007I: Resource bundles successfully removed from repository.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1008I: Cloudscape user functions file will be removed from repository if present.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1009I: Removing Cloudscape user functions from repository.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1010I: Cloudscape user functions successfully removed from repository.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1011I: Application Manager appmgr found. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1012I: Checking for installed UDDI Registry application of name appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1013W: Application of name appname is not present. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1014I: ApplicationManager not running, so application will not need to be stopped.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1015I: Stopping application of name appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1016W: stopApplication command for application appname caught exception Exc. Application might not have been running on this server. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1017I: Application appname stopped successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1018I: Removing application appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1019I: Application appname removed successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1020I: UDDI datasource will be removed from server servername in node nodename if present. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1021I: Removing UDDI datasource.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1022I: UDDI datasource successfully removed.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1023I: UDDI JDBC driver will be removed from server servername in node nodename if present. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1024I: Removing UDDI JDBC driver.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1025I: UDDI JDBC driver successfully removed.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1026I: UDDI configuration properties file will be removed from repository if present.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1027I: Removing configuration properties file from cell cellname, node nodename and server servername. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1028I: Configuration properties file successfully removed from repository.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1029I: Issuing nodeSync.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1030I: UDDI Registry application, JDBC driver and datasource removed successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1031I: Restarting application server.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1032I: Application server servername restarted successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1033I: Please ignore any errors concerning the serverStartupSyncEnabled attribute.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1034I: ws.ext.dir processing starting.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1035I: serverID is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1036I: JVM is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1037I: Out of N properties we located M matches at positions poslist. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1038I: Removing UDDI values from ws.ext.dirs properties.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1039I: ws.ext.dir has been set with new sysprop. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1040I: ws.ext.dir update skipped, required changes already present.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1041I: ws.ext.dir processing step complete.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1101I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1102I: Changes saved successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1103I: Attempting to save new configuration.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1104I: Changes saved successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1105I: Attempting to save new configuration.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1106I: Changes saved successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1107I: Attempting final save of new configuration.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1108I: Changes saved successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1109I: Attempting to save ws.ext.dir changes.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN1110I: Changes saved successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script removeuddi.jacl.

**User Response:** None.

**UDIN2001I: Assuming hard coded defaults.**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2002I: Listing members of type parenttype. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2003I: List for type parenttype returned N members. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2004I: Seeking parenttype with requested id of parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2005I: Checking parentID with parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2006I: Using this as parenttype of parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2007I: Checking for existing childtype under parentname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2008I: List for childtype returned N members. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2009I: No existing childtype present. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2010I: N existing objects of type childtype found, examining for conflict with childname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2011I: Checking childID with name childname. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2012I: Conflict found with existing childtype of id childID. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2013I: Not creating requested object of type childtype. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2014I: Conflict found with existing childtype, removing existing childtype. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2015I: Removal of childtype was successful. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2016I: Not in conflict.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2017I: Attempting to create childtype under parentname of parentID. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2018I: Create command that will be issued is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2019I: childtype childID was successfully created. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2020I: No matches found.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2021I: Looking for builtin_rra.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2022I: List for J2CResourceAdapter returned N members. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2023I: Hunting J2CResourceAdapter associated with Node nodename. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2024I: Using rraID as builtin_rra. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2025I: Using provider class of implclass with a classpath of clpath. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2026I: Installing to node nodename using database type of dbtype. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2027I: Attempting to create UDDI JDBCProvider.**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2028I: Attempting to create UDDI Datasource.**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2029I: Application Manager appmgr found. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2030I: Attempting to install UDDI Registry application.**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2031I: Checking for installed UDDI Registry application of name appname. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2032I: List for Applications returned N members. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2033W: Application of name appname is not present. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2034I: ApplicationManager not running, so application will not need to be stopped.**
**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2035I: Stopping application of name appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2036W: stopApplication command for application appname caught exception Exc. Application might not have been running on this server. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2037I: Application appname stopped successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2038I: Removing application appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2039I: Application appname removed successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2040I: Attempting to install application appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2041I: Starting UDDI application.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2042I: Application appname started successfully. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2101I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2102I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2103I: Changes were not saved on this call.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2104I: Attempting to save post installation configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2105I: Changes saved successfully for UDDI Registry.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2106I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN2107I: Changes saved successfully for UDDI Registry.**

**Explanation:** This is an informational message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:** None.

**UDIN3001I: Application Manager appmgr found. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3002I: Checking for installed UDDI Registry application.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3003I: List for Applications returned N members. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3004W: Application of name appname is not present. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3005I: ApplicationManager not running, so application will not need to be stopped.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3006I: Stopping application of name appname. Value is:**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3007W: stopApplication command for application appname caught exception Exc. Application might not have been running on this server. Values are:**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3008I: Application appname stopped successfully. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3009I: Removing application appname. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3010I: Application appname removed successfully. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3011I: UDDI datasource will be removed from server servername in node nodename if present. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3012I: Removing UDDI datasource.**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3013I: UDDI datasource successfully removed.**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3014I: UDDI JDBC driver will be removed from server servername in node nodename if present. Values are:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3015I: Removing UDDI JDBC driver from node nodename. Value is:**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3016I: UDDI JDBC driver successfully removed.**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3017I: UDDI Registry application, JDBC driver and datasource removed successfully.**
**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3101I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3102I: Changes to remove UDDI Registry have been saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3103I: Attempting to save new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3104I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3105I: Attempting final save of new configuration.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN3106I: Changes saved successfully.**

**Explanation:** This is an informational message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:** None.

**UDIN6001E: This script must be run in a Deployment Manager environment.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6002E: To install in a standalone application server, use appserversetupuddi.jacl.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6003E: Incorrect number of arguments passed to script.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6004E: Usage is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl. The text following 'Usage is:' gives the syntax for calling setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6005E: (<db2userid> <db2password> <db2ziplocation> are only required if setting up to use DB2).**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6006E: Use all forward ('/') slashes to avoid problems with escaping back ('\\') slashes.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6007E: Removal of childtype childname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6008E: An exception Exc occurred while creating childtype. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6009E: Unable to find requested parentype of parentname. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6010E: List command for J2CResourceAdapter caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6011E: No J2CResourceAdapter objects available.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6012E: An error occurred during execution of setupuddi.jacl. Please check the parameters and try again.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6013E: Uninstall of application appname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6014E: Install of UDDI application caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6015E: Could not get JVM.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6016E: Cannot find nodeSync MBean.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6017E: nodeSync failed. UDDI Application may not be fully installed.**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6018E: stopServer command for server servername caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6019E: startServer command for server servername caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6101E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6102E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6103E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN6104E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script setupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7001E: This script must be run in a Deployment Manager environment.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7002E: To remove from a standalone application server, use appserverremoveuddi.jacl.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7003E: Incorrect number of arguments passed to script.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7004E: Usage is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl. The text following 'Usage is:' gives the syntax for calling removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7005E: stopServer command for server servername caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7006E: Removal of resource bundles caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7007E: Removal of Cloudscape user functions caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7008E: Uninstall of application appname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7009E: Removal of UDDI datasource caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7010E: Removal of UDDI JDBC driver caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7011E: Removal of configuration properties file caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7012E: Cannot find nodeSync MBean.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7013E: nodeSync failed. UDDI Application may not be fully uninstalled.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7014E: startServer command for server servername caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7015E: Could not get JVM.**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7101E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7102E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7103E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7104E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN7105E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script removeuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8001E: This script must be run on a standalone application server.**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8002E: To install in a Deployment Manager Environment, use setupuddi.jacl.**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8003E: Incorrect number of arguments passed to script.**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8004E: Usage is:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl. The text following 'Usage is:' gives the syntax for calling appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8005E: (<db2userid> <db2password> <db2ziplocation> are only required if setting up to use DB2).**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8006E: Use all forward ('/') slashes to avoid problems with escaping back ('\\') slashes.**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8007E: List command for type parenttype caught exception Exc. Values are:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8008E: No objects of type parenttype available. Value is:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8009E: List command for childtype caught exception Exc. Values are:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8001E: This script must be run on a standalone application server.**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8010E: Error during remove of existing childtype, exception Exc caught. Values are:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8011E: Create command for childtype caught exception Exc. Values are:**
**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8012E: Unable to find requested parentype of parentname. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8013E: List command for J2CResourceAdapter caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8014E: No J2CResourceAdapter objects available.**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8015E: An error occurred during execution of appserversetupuddi.jacl. Please check the parameters and try again.**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8016E: List command for Applications caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8017E: Uninstall of application appname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8018E: Install of UDDI application caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8019E: startApplication command for appname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8101E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8102E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN8103E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserversetupuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9001E: This script must be run on a standalone application server.**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9002E: To remove from a deployment manager environment, use removeuddi.jacl.**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9003E: Incorrect number of arguments passed to script.**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9004E: Usage is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9005E: List command for Applications caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9006E: Uninstall of application appname caught exception Exc. Values are:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9007E: Removal of UDDI datasource caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9008E: Removal of UDDI JDBC driver caught exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9101E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9102E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

**UDIN9103E: Error saving configuration, changes not saved due to exception Exc. Value is:**

**Explanation:** This is an error message issued by the UDDI setup script appserverremoveuddi.jacl.

**User Response:**This message is self-explanatory.

## UDLC (Web Services UDDI) messages

There are no messages issued by this component.

## UDPR (Web Services UDDI) messages

There are no messages issued by this component.

## UDRS (Web Services UDDI) messages

**UDRS0001E: Exception ″<exception>″ occurred while attempting to get UDDI Message Logger.**

**Explanation:** This message is issued to stderr when an attempt to get the UDDI Message Logger fails with the indicated exception. Since the attempt to get the message logger failed, the message cannot be logged. No messages can be logged by this instance of the IBM WebSphere UDDI Registry.

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDRS0002E: Exception ″<exception>″ occurred while attempting to get UDDI Trace Logger for ″<component>″.**

**Explanation:** This message is logged when an attempt to get the UDDI Trace Logger for the specified component (or package) fails with the indicated exception. No trace entries can be logged for this component or package of the IBM WebSphere UDDI Registry.

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

## UDSC (Web Services UDDI) messages

There are no messages issued by this component.

## UDSP (Web Services UDDI) messages

**UDSP0001E: ParserPool found empty whilst attempting to process request. Request unsatisfied**

**Explanation:** A SOAP request was received, but was unable to be dealt with, as there were no free Parsers within the ParserPool.

**User Response:** Consider increasing the number of Parsers within the ParserPool by modifying the Init Parameter on the SOAP servlets.

**UDSP0002E: Error locating schemas required for UDDI processing. SOAP Servlets unworkable.**

**Explanation:** The SOAP servlet was unable to locate the schemas it requires in order to process SOAP requests. Without these, the servlet cannot process SOAP requests.

**User Response:** Check installation of UDDI was performed correctly. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDSP0003W: Servlet unable to locate init parameter 'defaultPoolSize'. Using internal defaults.**

**Explanation:** The SOAP servlet was unable to locate the init parameter which sets the default size of the ParserPool. It will fall back to an internal default.

**User Response:** If this message occured after attempting to make changes to the defaultPoolSize init parameter, ensure the changes were correct. If this message has appeared after installed, ensure installation was performed correctly.

**UDSP0004W: Servlet unable to understand init parameter 'defaultPoolSize'. Using internal defaults.**

**Explanation:** The SOAP servlet was unable to parse the init parameter which sets the default size of the ParserPool. It will fall back to an internal default.

**User Response:** If this message occured after attempting to make changes to the defaultPoolSize init parameter, ensure the changes were correct. If this message has appeared after installed, ensure installation was performed correctly.

**UDSP0005E: Error occurred during parser creation.**

**Explanation:** An unspecified error occured during the creation of a SOAP parser

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDSP0006E: Internal configuration error.**

**Explanation:** This error may occur if there was a failure creating a Parser, with accompanying message UDSP0005. It may also occur if there was a problem acquiring the Persistence layer.

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDSP0007E: Error during servlet acquisition of persistence layer.**

**Explanation:** The SOAP servlet was unable to acquire the persistence layer required for it to communicate with the UDDI datasource

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDSP0008E: Error during servlet release of persistence layer.**

**Explanation:** The persistence layer reported a problem when the SOAP servlet attempted to release it.

**User Response:** Restart the UDDI registry. If the error persists, examine the WebSphere logs for information on its cause. If the problem cannot be resolved, then please contact the IBM Customer Service Center.

**UDSP0009E: Error during sending of response to client.**
**Explanation:** An error occured when sending a SOAP response message back to a client. The client may not have received the response

**User Response:** This error is recorded to enable logging of failed responses to clients. The error may be the fault of the client disconnecting before the reply could be sent, or may indicate a network problem. Examine the WebSphere logs for more information on its cause.

## UDUC (Web Services UDDI) messages

**UDUC0001I: IBM WebSphere UDDI Registry user console starting initialization.**
**Explanation:** The user console control servlet is starting.

**User Response:** None.

**UDUC0002I: IBM WebSphere UDDI Registry user console finished initialization.**
**Explanation:** The user console control servlet has completed startup.

**User Response:** None.

**UDUC0003I: Reading init parameters.**
**Explanation:** The user console control servlet has started reading external parameters in its init method

**User Response:** None.

**UDUC0004I: Finished reading init parameters.**
**Explanation:** The user console control servlet has finished reading external parameters in its init method. This message indicates the user console is ready to accept client requests.

**User Response:** None.

**UDUC0005E: A serious error has occurred. Error message: <Message> error: <Throwable>. More information: <Additional information>.**
**Explanation:** This error message indicates an unexpected error has occurred. The <Message> describes the error that has occurred and the <Throwable> is the type of error that was caught. <Additional information> may provide further information, if available.

**User Response:** A trace of the gui component is recommended. Contact IBM support with this information.

**UDUC0006E: A persistence error has occurred. Error message: <Message> error: <Throwable>. More information: <Additional information>.**
**Explanation:** An error occurred while performing a database operation. The <Message> describes the error that occurred and the <Throwable> is the type of error that was caught. <Additional information> may provide further information, if available.

**User Response:** Check database connections and state. Please provide IBM support with a trace, including the gui and persistence components.

**UDUC0007E: A User mismatch error has occurred. Error message: <Message> error: <Throwable>. More information: <Additional information>.**
**Explanation:** The user id provided does not match the user id required or expected whilst performing an operation that requires authentication. The

<Message> describes the error that occurred and the <Throwable> is the type of error that was caught. <Additional information> may provide further information, if available.

**User Response:** Check the user has authority for the operation being requested. If necessary, contact IBM support detailing the actions taken to recreate the problem.

**UDUC0008E: An invalid key was passed. Error message: <Message> error: <Throwable>. More information: <Additional information>.**

**Explanation:** The requested operation is trying to retrieve information about an entity with a key that is invalid. This may occur if the entity has been deleted by another session. The <Message> describes the error that occurred and the <Throwable> is the type of error that was caught. <Additional information> may provide further information, if available.

**User Response:** Ask the client to close existing sessions and attempt the operation in a new browser session. If the problem persists, please provide IBM support with a trace of the gui and api components.

**UDUC0009E: An invalid value was supplied. Error message: <Message> error: <Throwable>. More information: <Additional information>**

**Explanation:** An invalid value was passed to an API call. The >Message> describes the error that occurred and the <Throwable> is the type of error that was caught. <Additional information> may provide further information, if available.

**User Response:** Contact IBM support with a trace of the gui and api components.

**UDUC0010E: Failed to introspect ActionForm properties. Exception: <Exception>.**

**Explanation:** String properties of a form object could not be introspected which means that the form contents cannot be checked for invalid characters.

**User Response:** Please contact IBM support with details of the Exception and a trace of the gui component.

**UDUC0011E: Failed to invoke reflected methods in ActionForm. Exception: <Exception>.**

**Explanation:** A form object's declared public method for setting or getting a String value could not be invoked. This method is required to check for invalid characters.

**User Response:** Please contact IBM support with details of the Exception and a trace of the gui component.

**UDUC0012E: User console initialization failed to connect to UDDI database. Exception: <Exception>.**

**Explanation:** During user console initialization, connection to the database failed, and threw the exception specified.

**User Response:** Check the connection to the UDDI database. The included exception message may yield some clues to help you resolve the problem. If unresolved, please contact IBM support with a trace of the gui component during startup.

**UDUC0013E: User console initialization failed to initialize tModels. Exception: <Exception>.**

**Explanation:** Indicates that an error has occurred during initialization of ActionServlet, specifically when reading tModels (invoking init method in class TModelNames).

**User Response:** Check the state of the UDDI database. Visually inspect the TMODEL table and confirm it is populated with valid data. The included exception message may yield some clues to help you resolve the problem. If unresolved, please contact IBM support with a trace of the gui component during startup.

**UDUC0014E: User console initialization failed to initialize taxonomies. Exception: <Exception>.**

**Explanation:** Indicates that an error has occurred during initialization of ActionServlet, specifically when reading taxonomy data (invoking init method of CategoryTaxonomyTree).

**User Response:** Check the state of the UDDI database. The included exception message may yield some clues to help you resolve the problem. If unresolved, please contact IBM support with a trace of the gui component during startup.

### UDUU (Web Services UDDI) messages

There are no messages issued by this component.

## Running the UDDI Samples

The UDDI samples, and documentation on how to use them, are available through the Web Services UDDI samples link on the

(http://www.ibm.com/websphere/developer/library/samples/AppServer.html) page of the IBM WebSphere Developer Domain Web site.

## Installation Verification Program (IVP)

Before you begin

There are some samples available on the WSDD web site that are intended to provide an optional Installation Verification test, or IVP, for the UDDI Registry component.

This topic describes how to run these installation verification programs (IVPs) to verify that the IBM UDDI Registry has been installed correctly.

There are two IVP SOAP samples called SOAPSampleIVPa and SOAPSampleIVPb. They are intended to verify the successful installation of the product, and should be used in conjunction with the UDDI Users Console (GUI). SOAPSampleIVPa saves some data to the registry which you can then find using the GUI. Finally you can delete the data by running SOAPSampleIVPb.

The IVP samples are installed into the same target directory as the other SOAP samples and they use the same XML files as the basic Java SOAP samples.

SOAPSampleIVPa saves three businesses, six services (2 per business) and three tModels. The data structures are very basic and consist only of a name. The keys returned by the save_* UDDI API calls are then written to a file, `SOAPSampleIVPa.out`. SOAPSampleIVPb then reads in these keys from the file in order to delete the saved data from the UDDI registry.

**Note:** Each time you run SOAPSampleIVPa, it overwrites the output file `SOAPSampleIVPa.out` so, if you wish to use SOAPSampleIVPb to delete the data, you must run this before you next run SOAPSampleIVPa.

To run the IVPs, complete the following steps on the same system as the UDDI Registry:

1. Ensure that DB2 and the WebSphere Admin Server are started.
2. Start the WebSphere Administrator's Console and ensure the default server is started and the UDDI Registry Application is started.
3. For SOAP samples to work, you need to ensure that the Client JDK is either the one shipped with IBM WebSphere Application Server or a later IBM JDK.:
   - For Windows - ensure that <WebSphere-install-dir>\java\bin is present in the PATH statement before any other JDK's
   - For Unix Platforms - ensure that <WebSphere-install-dir>/java/bin is present in the PATH statement before any other JDK's

   **Note:** You **must** use the IBM WebSphere supplied JDK or a later level of the IBM JDK.

   For Windows, the default system path can be set via **Control Panel ...-> Settings ...-> System ...-> Advanced Properties ...-> Environment Variables**

   Alternatively, this can be accomplished just for the shell where you plan to run the samples by modifying the path within the shell:
   - For Windows - set path=<WebSphere-install-dir>\java\bin;%path%
   - For Unix Platforms - export PATH=<WebSphere-install-dir>/java/bin:$PATH
4. Copy the samples and *.xml files to a directory, and compile and run them there (see next bullets)
5. Compile both SOAPSampleIVPa and SOAPSampleIVPb by typing:
   ```
   'javac SOAPSampleIVPa.java'
   ```
   and
   ```
   'javac SOAPSampleIVPb'
   ```
   .
6. Run SOAPSampleIVPa by typing 'java SOAPSampleIVPa'. This should publish a number of businesses and services and technical models into the registry.
7. Start your Web browser on the same system as the UDDI Registry.
8. To display the UDDI GUI home page, type one of the following URLs:
   - If you have WebSphere security disabled: `http://localhost:9080/uddigui`
   - If you have WebSphere security enabled: `https://localhost:9433/uddigui`
9. On the find page, complete the following steps:
   a. Select the business radio button
   b. In the data entry field, type `%` (percent is the wild card symbol)
   c. Click **Find**

You should get a results page returned with three businesses (mybusiness1, mybusiness2, and mybusiness3). This demonstrates that the API and the UDDI Console are working correctly.

10. To see the services that are available for a business, click the ″Show Services″ option next to the business.
11. To delete all of the IVP data, run SOAPSampleIVPb (from the command prompt as before - by typing ’java SOAPSampleIVPb’)
12. On the find page, complete the following steps:
    a. Select the business radio button
    b. In the data entry field, type `%` (percent is the wild card symbol)
    c. Click **Find**

       You should get an empty results page returned.

## Reporting Problems with the IBM WebSphere UDDI Registry

Before you begin

If you report a problem with the IBM WebSphere UDDI Registry component to IBM, please supply the following information:

1. A detailed description of the problem.
2. The build date and time of the version you are using. This can be obtained as follows:
   - In the installedApps subdirectory of the WebSphere install location, you will find a subdirectory called UDDI_Registry.<nodename>.<servername>.ear, where <nodename> is the name of the node into which the UDDI Registry application is installed, and <servername> is the name of the server. Within that subdirectory, you will find a file called version.txt. Please include the contents of this file as part of your information.
   - If the UDDI Registry has been started with tracing enabled for the UDDI component, then you should find a trace entry in the WebSphere trace log which includes the strings ″ getUDDIMessageLogger″ and ″UDDI Build :″ followed by the build date and time, and the build system. Please also include this information.
3. Any relevant log files and trace files.
   - If the problem occurred while setting up and installing the UDDI Registry application using one of the setup scripts, setupuddi.jacl or appserversetupuddi.jacl, then please supply the log output from running the script. (If you had not chosen to redirect the output from the script file to a log file, then please rerun the script, this time redirecting the output as described in the section ’Installing and Setting up a UDDI Registry’.) The log file will be in the directory from which you ran the setup script.
   - If the problem occurred while removing the UDDI Registry application using one of the remove scripts, removeuddi.jacl or appserverremoveuddi.jacl, then please supply the log output from running the script. (If you had not chosen to redirect the output from the script file to a log file, then please rerun the script, this time redirecting the output as described in the section ’Removing the UDDI Registry from a deployment manager cell’ or ’Removing the UDDI Registry application from a single appserver’.) The log file will be in the directory from which you ran the remove script.
   - If the problem occurred while creating the UDDI Registry database using the UDDI DB2 Setup Wizard, then please supply the log file UDDIloadDB.log, which will be in the directory from which the wizard was run.

- If the problem occurred while running the UDDI Registry, please enable UDDI tracing (if not already enabled) and supply the trace log from the logs directory of the application server on which the UDDI Registry was running. Please refer to the section on ′Turning on UDDI Trace′ for details on how to enable UDDI tracing.

4. If appropriate, any application code that you are using and the output produced by the application code.

## Feedback

Before you begin

See the section on ″Obtaining help from IBM″ elsewhere in this InfoCenter for details on seeking assistance.

# Chapter 6. Enabling Web services through the IBM Web Services Gateway

Use this topic to enable Web services through the IBM Web services Gateway

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 registrations.

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″.
- ″Web Services Gateway troubleshooting tips″.

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 additional technical details of the Web Services Gateway, see the gateway (../javadoc/wsg/index.html).

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 Definition Language (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 which 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, or a SOAP 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 (http://www.alphaworks.ibm.com/tech/wsgw) on 21 December 2001. The main differences between the AlphaWorks edition and the current 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, as well as for the gateway as a whole.

## Web Services Gateway - Completing the installation

Before you begin

This task assumes that, when you installed WebSphere Application Server Network Deployment, 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 *`WebSphere_DeployMgr_root`*, 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:\Progra~1\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 |

Before you can run the Web Services Gateway, you need to complete the installation. You have two choices:
- Install into a deployment manager cell.
- Install into a standalone application server.

For production use you will probably want to install the gateway into a deployment manager cell, but for development or test purposes you might find it useful to install the gateway into a standalone application server.

What to do next

To finish the Web Services Gateway installation, complete the following steps:
- Confirm that your system configuration complies with the Web services Gateway prerequisites and constraints.
- Establish requirements for using a database with the gateway.
- Enable security if required.
- *either* Install the gateway into a deployment manager cell.
- *or* Install the gateway into a standalone application server.
- Test the installation.

## Web Services Gateway - prerequisites and constraints

The Web Services Gateway assumes that IBM WebSphere Application Server Advanced Edition (AE) Version 5.0, or IBM WebSphere Application Server Advanced Edition - Developer Only Option (AEd) Version 5.0, and any prerequisites they require, are installed on your system.

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.
- The gateway does not support WSDL service definitions that contain **soap:header** elements within their **wsdl:definition** element.
- When you deploy a Web service to the gateway, the WSDL file that describes the Web service must be located at a URL. It cannot be located through UDDI lookup information.
- You must publish your Web services manually to any UDDI registries.

You might also find it useful to enable trace for all Gateway components, 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 5.0.3 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 standalone 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, and that you are installing into an existing Deployment Manager cell.

If you want to enable gateway-level security, you must do so before you install the Web Services Gateway and SOAP channel 1.

In this task you install the gateway into 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.
  - SOAP channel 1.
  - The correlation application (optional - only if a database is being installed).
- Installation of other gateway applications. For example SOAP channel 2.

To install the gateway into 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 `*WebSphere_DeployMgr_root*/WSGW/scripts/install`, where *WebSphere_DeployMgr_root* is the Deployment Manager root directory (by default `WebSphere/DeploymentManager`).

b. Enter the command `WSGWinstallDB.ext WebSphere_DeployMgr_root db2user_id db2password`

   where *.ext* is `.bat` for a Windows system and `.sh` for a UNIX system. For example (UNIX systems):

   ```
   ./WSGWinstallDB.sh /opt/WebSphere/DeploymentManager mydb2id mydb2pw
   ```

   **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.ext your_server`).
3. From a command prompt, go to directory `*WebSphere_DeployMgr_root*/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 `WebSphere_DeployMgr_root/bin/wsadmin.ext -f setupWSGW.jacl parm1 ... parmN`

   where
   - *parm1* is the *WebSphere_DeployMgr_root* 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.jacl 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 (`*WebSphere_DeployMgr_root*/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.

6. The gateway (`wsgw.ear`), SOAP channel 1 (`wsgwsoap1.ear`) and (optionally) correlation (`wsgwcorr.ear`) applications have been installed by `setupWSGW.jacl` in the previous steps. To install additional Web services Gateway applications - for example SOAP channel 2 (`wsgwsoap2.ear`) - 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.

   a. From a command prompt, go to directory `*WebSphere_DeployMgr_root*/bin`
   b. To start the application server, enter the command `startServer.*ext* *your_server*`
   c. To start the WebSphere administration program, enter the command `wsadmin.*ext*`.
   d. For each additional Web Services Gateway enterprise application that you want to install, enter the following commands (shown split for publication) 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, shown split for publication):

      ```
      wsadmin>$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.*ext* *your_server*`
   b. Enter the command `startServer.*ext* *your_server*`

What to do next

You are now ready to test the installation.

## Installing the gateway into a standalone 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.

**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 synchronisation process that takes place within a cell of application servers.

When you installed WebSphere Application Server Network Deployment, 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.`*`ext your_server`*

  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 *`standalone_WAS_root`*`/installableApps` directory of the target application server's install tree, where *standalone_WAS_root* is the root directory of your target application server (by default `WebSphere/AppServer`).
- Copy file *`WebSphere_DeployMgr_root`*`/lib/WSGWResouceBundles.jar` into directory *`standalone_WAS_root`*`/lib`.
- Copy directory *`WebSphere_DeployMgr_root`*`/WSGW` and all files and directories within it into directory *`standalone_WAS_root`*`/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 *`standalone_WAS_root`*`/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.

  - SOAP channel 1.
  - The correlation application (optional - only if a database is being installed).
- Installation of other gateway applications. For example 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 `standalone_WAS_root/WSGW/scripts/install`.
   b. Enter the command `WSGWinstallDB.ext standalone_WAS_root db2user_id db2password`

      For example (UNIX systems): `./WSGWinstallDB.sh /opt/WebSphere/AppServer mydb2id mydb2pw`

      **Note:** Running `WSGWinstallDB` also creates `WSGWInstallDB.log` in the standalone 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.ext your_server`).
3. From a command prompt, go to directory `standalone_WAS_root/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 `standalone_WAS_root/bin/wsadmin.ext -f setupWSGW.jacl parm1 ... parmN`

   where
   - *parm1* is the `standalone_WAS_root` 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.jacl
   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 (`standalone_WAS_root/bin/WSGWDB`).

   For example (Windows systems, shown split for publication):

```
C:\Progra~1\WebSphere\AppServer\bin\wsadmin.bat -f setupWSGW.jacl
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.jacl
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.

6. The gateway (`wsgw.ear`), SOAP channel 1 (`wsgwsoap1.ear`) and (optionally) correlation (`wsgwcorr.ear`) applications have been installed by `setupWSGW.jacl` in the previous steps. To install additional Web services Gateway applications - for example SOAP channel 2 (`wsgwsoap2.ear`) - 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.

   a. From a command prompt, go to directory `*standalone_WAS_root*/bin`
   b. To start the application server, enter the command `startServer.*ext* *your_server*`
   c. To start the WebSphere administration program, enter the command `wsadmin.*ext*`.
   d. For each additional Web Services Gateway enterprise application that you want to install, enter the following commands (shown split for publication) 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, shown split for publication):

   ```
   wsadmin>$AdminApp install
    C:/Progra~1/WebSphere/AppServer/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.`*`ext your_server`*
   b. Enter the command `startServer.`*`ext your_server`*

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://`*`host`*`:`*`port`*`/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
2. If the previous step was successful, then to test the Apache SOAP channels use your Web browser to display the Web page at `http://`*`host`*`:`*`port`*`/wsgw`*`engine`*`/soaprpcrouter` where *engine* is either `soap1` or `soap2`.

   The browser should display the following message: `Sorry, I don't speak via HTTP GET - you have to use HTTP POST to talk to me.`

What to do next

If you don't see these messages, your server is not configured correctly - in which case, see Web Services Gateway troubleshooting tips.

## Administering the Web Services Gateway

Use this task to administer the Web Services Gateway

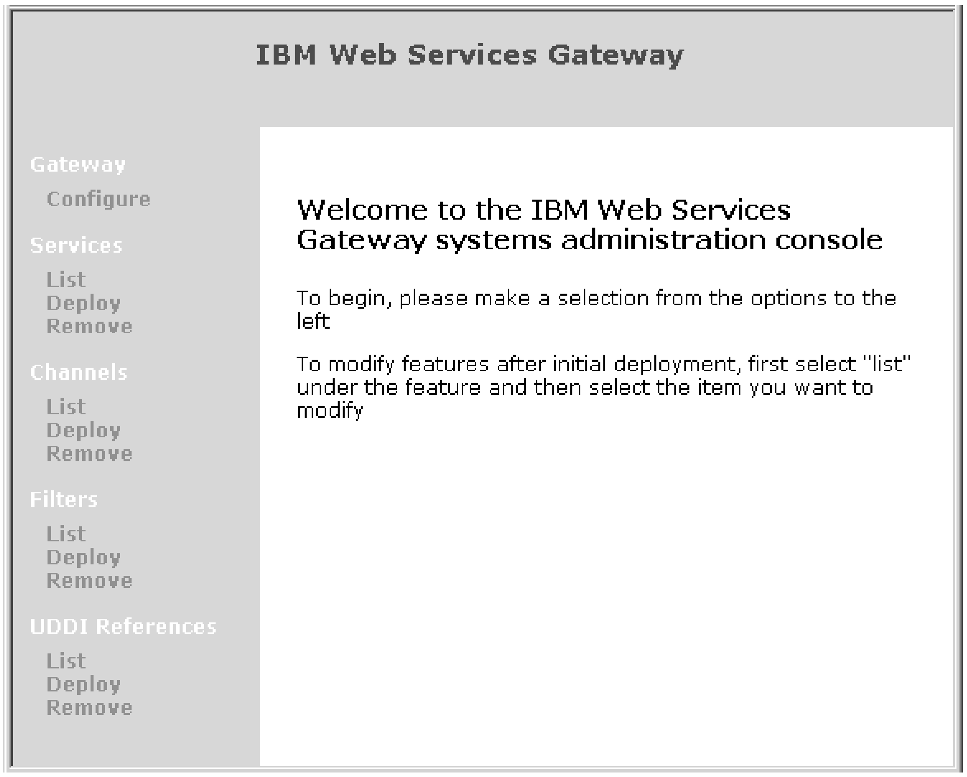
To administer the Web Services Gateway, complete the following steps:

Steps for this task

1. Start the WebSphere Application Server Administrative Server.
2. Open the following Web page: `http://`*`host`*`:`*`port`*`/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:



Welcome to the IBM Web Services Gateway systems administration console

To begin, please make a selection from the options to the left

To modify features after initial deployment, first select "list" under the feature and then select the item you want to modify

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

Use this task to set the namespace 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:

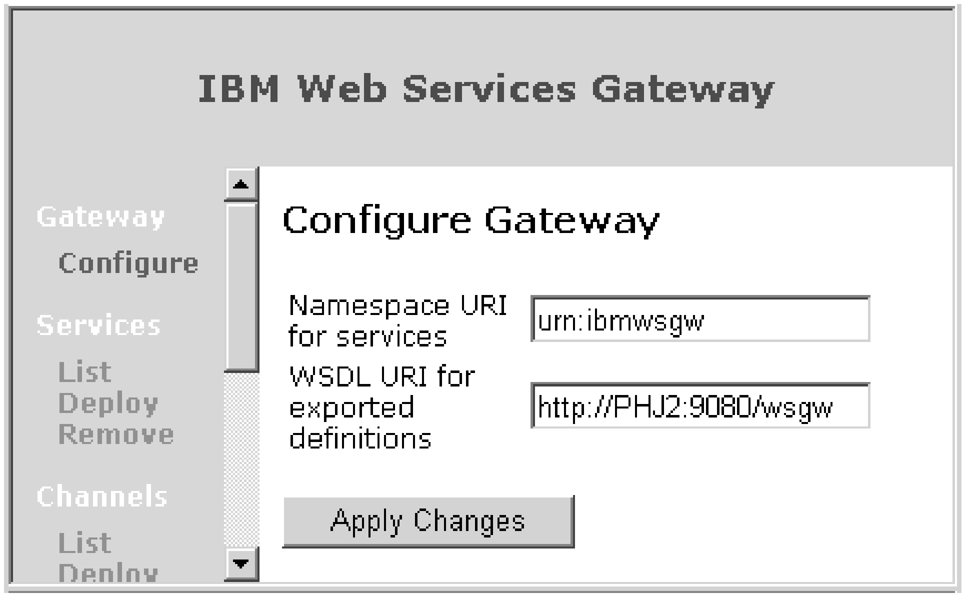
Steps for this task

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:

   Configure Gateway

   Namespace URI for services: urn:ibmwsgw
   WSDL URI for exported definitions: http://PHJ2:9080/wsgw

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).
- 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://hldswrth:9080/wsgw`, and because you are giving the WSDL to people in other companies you modify this to `http://hldswrth.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**.

## Working with channels

Use this task to administer 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 standalone 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 organisation from the rights you give to users within your organisation:

- To ensure that users outside your organisation 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 organisation 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

Use this task to deploy a channel 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 standalone 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 Web Services 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
  - **End Point Address:** http://*host*:*port*/wsgwsoap1

    where *host* and *port* are the host name and port number for the application server on which the channel application is installed.
  - **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
  - **End Point Address:** http://*host*:*port*/wsgwsoap2
  - **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

Use this task to remove a channel 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

Use this task to administer 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 standalone 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 standalone 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

Use this task to administer 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. For more information about UDDI and UDDI Registries, see the UDDI community at (http://uddi.org).

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.

### 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 that has update access to the UDDI registry.

**Password**
Type the password for the new user ID specified in the **User Name** field.

**Confirm Password**
Type the password for the new user ID specified in the **User Name** field. This must match the password specified in the **Password** field.

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

Use this task to deploy a UDDI reference to the Web Services Gateway.

**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 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 that has update access to the UDDI registry.

   **Password**
   Type the password for the user ID specified in the **User Name** field.

**Confirm Password**

Type the password for the user ID specified in the **User Name** field. This must match the password specified in the **Password** field.

**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 .

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

Use this task to configure a Web service.

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.
- Any filters that apply to the service.
- Any UDDI references to UDDI registries in which the service is deployed.

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 UDDI references 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`

5. In the **Target Services** section, you can add or remove single instances of multiple target services for this gateway service. To add a new target service instance, complete the following steps:

   a. **WSDL Location**. Type the location of the ″internal″ WSDL file that describes the Web service to be deployed. This value is either a URL, or (if the WSDL is located in a UDDI Registry) 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`

      **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.

   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, 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, 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.

      **Note:** If you are mapping multiple target services, and also , you might use the Target Service Identity Information to select a particular target service from the set.

   f. Click **add**.

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.
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 (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

Use this task to deploy a Web service to the Web Services Gateway.

Before you begin

Before you deploy a Web service, deploy the resources (channels, filters, and UDDI references) that the Web service uses.

For Web services deployed with Java bindings (or EJB bindings where the Web service is on a different server), make the additional classes available to the gateway as described in Deploying Web services with Java bindings.

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 Web Services Gateway (and by which it will be listed using the Services > List option). This name must be unique within the Web Services 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**:

   **Generic classes**

   Select this option if there are no EJB or Java bindings, or if there are such bindings but they are not used at runtime.

   **Deployed Java classes**

   Select this option if the target services for the gateway service contain EJB bindings or Java bindings.

   **Note:** When you choose this option, you must also ensure that the specific java classes that have been generated for the Web service are deployed to the application server (for example by copying the JAR file into the WebSphere Application Server's `/AppServer/lib/app` directory, then restarting the application server).

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**

   In this field you can select zero or more deployed UDDI references to UDDI registries in which this service is to be registered. If you select one or more UDDI references in this step, you must also enter a UDDI business key in step 3h below.

   **Note:** There may be more than one UDDI Reference for the same business in the same UDDI Registry. This is because a business can have more than one entry (each with a different ″Authorized Name″) in the same UDDI registry. For more information see Deploying UDDI references to 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. This value is either a URL, or (if the WSDL is located in a UDDI Registry) 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`

**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.

**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, type the target service’s name from the target service WSDL.

**Target Service Namespace**
If the Web service WSDL contains more than one service, 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.

**Note:** If you later add more target services to this gateway service, and also , you might use the Target Service Identity Information to select a particular target service from the set.

h. If you want to publish the service to one or more UDDI registries (selected in step 3f above), 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`

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

After deployment, use the list deployed Web services option to change the resources (channels, filters, and UDDI references) that the Web service uses, or to add multiple target services for this gateway service.

If you enabled the check box ’Authorization Policy - Control access to this service, you must now complete the task Enabling Web service operation-level authorization.

**Deploying Web services with Java bindings:** For Web services deployed with Java bindings (or EJB bindings 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 `*WAS_root*/lib` or `*WAS_root*/lib/app` directory (where *WAS_root* 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 `*WAS_root*/lib` or `*WAS_root*/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.

### Removing Web services from the Web Services Gateway

Use this task to remove a Web service 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.

## Running the Web Services Gateway samples

Use this task to run the samples provided with the Web services Gateway.

There are two pre-built samples 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 (http://www.ibm.com/websphere/developer/library/samples/AppServer.html) page 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 (IBM WebSphere UDDI Registry), you should complete the following additional steps:

1. Publish the WSDL for each of these samples to UDDI. (For detailed information on how to do this, see 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.

## Administering security for the Web Services Gateway

The Web Services Gateway provides a basic authentication and authorization mechanism based upon the security features of WebSphere Application Server. Security can be applied at two levels, as described in the following topics:

1. Enabling gateway-level authentication.
2. Enabling Web service operation-level 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 requestor'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 standalone application server) you also install 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 requestor’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 requestors 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 standalone 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 Web service 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 *`your_webservice`*`.ear`, complete the following steps:
   a. Open a command prompt.
   b. Go to directory `/WSGW/scripts/auth`
   c. Enter the command `WSGWAuthGen` *`location your_webservice`*
      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 `WSGWAuthGen http://`*host*`:`*port*`/wsgw AddressBook` 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 *`your_webservice`*`.ear`. A message appears informing you of the name change.

*`your_webservice`*`.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 *`your_webservice`*`.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/`*`your_webservice`*`.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 *`your_webservice`*`.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/`*`your_webservice`*).
- 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 modifed copy of `wsgwauth.ear`.

l. Close the AAT.

3. To install the modifed 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.

### Web service 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 Lightweight Third Party Authentication (LTPA).

## 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 *`WAS_root`*`/java/jre/lib/ext` (where *WAS_root* is the root directory for your installation of IBM WebSphere Application Server).

2. Edit the security properties file *WAS_root*/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 Administative 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");
   ```

## 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. 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 Message reference for the Web Services Gateway.

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 IllegalArgument 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 URLException 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.

## 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

**2W_key=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:** Ensyre 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 (command shown split for publication):
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 messsage 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 messsage 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 messsage 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 messsage 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 messsage 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 messsage 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 messsage is used by the WSGWAuthGen command line utility.

**User Response:** No action required.

**WSGW0098I: Command Status :**

**Explanation:** General command status message. This messsage 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 messsage 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 messsage 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 messsage 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 messsage 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 messsage 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.

Because the concept of a Web services gateway is so new, there is (as yet) very little supplementary information about it.

- **Inside WebSphere Application Server 5** (http://advisor.com/doc/09808) . This article from WebSphere Advisor Magazine (August 2002) mentions the gateway as part of a general discussion of the new features in this version of WebSphere Application Server.
- **Web Services Gateway** (http://www.alphaworks.ibm.com/tech/wsgw) . The Web Services Gateway area on the IBM alphaWorks Web site. This area provides a discussion forum for early adopters.
- **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.