IBM® WebSphere® Application Server V7 contains key improvements to the way it interoperates with IBM WebSphere MQ as a Java™ Message Service (JMS) provider. This article provides an overview of the new WebSphere MQ messaging provider with particular focus on its new capabilities and the enhancements in WebSphere Application Server V7.

Introduction

Many organizations use IBM WebSphere MQ to provide enterprise messaging and IBM WebSphere Application Server to deploy their Java Platform, Enterprise Edition (Java EE) applications. When you use the WebSphere MQ messaging provider support in WebSphere Application Server (originally introduced in V5), it is possible to deploy Java EE applications that directly leverage the enterprise messaging capabilities of WebSphere MQ. This article provides an overview of the new WebSphere MQ messaging provider and related enhancements that have been added to WebSphere Application Server V7.

This article assumes a general familiarity with WebSphere Application Server and WebSphere MQ as well as the Java Message Service (JMS) API. See Related topics for some good introductory articles.

What is the WebSphere MQ messaging provider?

Java EE applications that use the WebSphere MQ messaging provider (hereafter referred as "MQ messaging provider") do so using the JMS API (see Related topics), although WebSphere MQ also supports a wide selection of other programming APIs and environments. This means that Java EE applications can use the MQ messaging provider to communicate with CICS applications, back end applications running on z/OS®, and other existing applications written in a variety of languages, to give but a few examples (Figure 1).
With the MQ messaging provider, Java EE applications deployed to make use of local or global transactions can enlist WebSphere MQ as a transactional resource. For example, a purchase management application could be notified asynchronously of the arrival of a "place account" message from WebSphereMQ. In response to the message, the application might update an inventory database and, if the customer has insufficient credit, send another WebSphere MQ message to an invoicing application (Figure 2).

Of course, it would be preferable if this example application could perform these actions such that all resources are updated as a single global transaction. In fact, it is possible to do this using the MQ messaging provider. In other words, you can coordinate WebSphereMQ along with other transactional resources, such as an IBM DB2® database, in the scope of a single global transaction.

To use the MQ messaging provider with a Java EE application, you must define one or more administered objects. Administered objects are held in the application server's Java Naming and Directory Interface (JNDI) namespace and are used to relate an application using the JMS API to a resource defined to WebSphereMQ. As WebSphereMQ messaging provider administered objects are held in JNDI, they are accessible from any process with access to the JNDI namespace.

There are three categories of administered objects that are supported directly by the MQ messaging provider in WebSphere Application Server V7. These are:
• Activation specifications  
• Connection factories  
• Destinations.

Additionally, it is possible to configure message listener ports to use the MQ messaging provider. MQ messaging provider administered objects are discussed in more detail in the next sections.

**Activation specifications**

The updated MQ messaging provider support in WebSphere Application Server V7 is based on a J2EE™ Connector Architecture (JCA) 1.5 compliant resource adapter. Part of the JCA 1.5 specification defines a vendor-neutral way of enabling inbound messages to be delivered to message-driven beans (MDBs) running inside a Java EE compliant application server, such as WebSphere Application Server. The construct that is used to configure and setup inbound message delivery is known as an **activation specification**. As such, in WebSphere Application Server V7, it is now possible to create MQ messaging provider activation specifications to manage the relationship between an MDB running in WebSphere Application Server and a destination in WebSphere MQ.

MQ messaging provider activation specifications are now the preferred mechanism for delivering messages from a WebSphere MQ destination to an MDB running inside WebSphere Application Server. Activation specifications supersede the use of the existing WebSphere Application Server message listener port support, which has been deprecated as of WebSphere Application Server V7. However, it is still possible to use message listener ports to deliver messages to an MDB using the MQ messaging provider -- it is even possible to configure both message listener ports (which make use of MQ messaging provider resources) and MQ messaging provider activation specifications at the same time.

**Activation specifications versus message listener ports**

There are several reasons why it is preferable to use an activation specification rather than a message listener port to deliver messages to an MDB. Activation specifications in general (and MQ messaging provider activation specifications in particular) provide the following advantages over a message listener port:

- **Specification defined**: Activation specifications are part of a standards specification (JCA 1.5). Message listener port support in WebSphere Application Server makes use of the application server facilities interfaces defined in the JMS specification, but is not part of any specification itself.
- **Simple to configure**: To configure a message listener port, you need three configuration objects: a connection factory, a destination, and the message listener port itself. Using an activation specification requires only two objects: the activation specification and a destination.
- **Not limited to server scope**: Activation specifications can be defined at any administrative scope in WebSphere Application Server. Message listener ports have to be configured at server scope. This means that if you have a node made up of three servers, you must
configure three separate message listener ports, as opposed to only one activation specification.

**An activation specification in action**

Figure 3 shows how an MQ messaging provider activation specification can be used to link a WebSphere MQ queue manager destination to an MDB running inside WebSphere Application Server. The process of delivering a message from a client to an MDB via an MQ messaging provider activation specification occurs in this way:

1. A messaging client, either running in a standalone process or inside an application server environment, sends a message using JMS (or any other messaging API, such as MQI) to a WebSphere MQ queue or topic defined in a WebSphere MQ queue manager.
2. A WebSphere MQ activation specification is configured to listen on that destination for messages. When the new message is detected, it is removed from the destination (potentially under an XA transaction).
3. The message is then passed to an MDB that has been configured to use the MQ messaging provider activation specification through its onMessage method.
4. The MDB uses the information in the message to perform the relevant business logic.

**Figure 3. WebSphere MQ messaging provider activation specification in action**

Configuring activation specifications

Configuring MQ messaging provider activation specifications is very similar to configuring MQ messaging provider connection factories. They can be configured using either the WebSphere Application Server administrative console or the wsadmin command line tooling. To create an MQ messaging provider activation specification using the administrative console:

1. Log on to the administrative console.
2. Navigate to the Resources section. Expand the JMS tab, and click on Activation specifications.
3. Choose the scope at which you want to define the MQ messaging provider activation specification.
4. Click New.
5. Click the WebSphere MQ messaging provider box, and click OK.

Figure 4 is annotated with the locations of these key controls.
Once a basic MQ messaging provider activation specification has been created, it can be further configured using six different configuration panels. To further configure an MQ messaging provider activation specification:

1. Log into the administrative console.
2. Navigate to the **Resources** section, expand the **JMS** tab, and then click on **Activation specifications**.
3. Choose the scope at which the relevant WebSphere MQ messaging provider activation specification is defined.
4. Select the relevant MQ messaging provider activation specification.

The first panel provides the ability to modify the transport mode and destination of an MQ messaging provider activation specification, along with other commonly used configuration settings. The right side of this page provides links to the other configuration panels:

- The **Advanced properties** panel enables configuration of message compression, consumer settings, and message format settings. It also provides the ability to set the number of times an MQ messaging provider activation specification should attempt to try to deliver a message to an MDB, should that MDB repeatedly throw an exception. If the number of deliveries is exceeded and the activation specification is appropriately configured, the failing MDB is paused, providing an opportunity for the failure to be investigated.
- Settings used for performing pub/sub messaging can be configured using the **Broker properties** panel. Not all properties are appropriate for all versions of queue manager. More information about which properties apply with a particular version of queue manager is specified in the **WebSphere MQ Information Center**.
- The **Custom properties** panel can be used to enter any generic configuration settings for the MQ messaging provider activation specification in the form of name-value pairs. Typically, this information will not be necessary.
• Additional information on SSL settings and channel exits can be entered using the **Client transport properties** panel. This page will not be available if a client channel definition table (CCDT) based MQ messaging provider activation specification has been created.

• Finally, the **JAAS-J2C authentication data** panel can be used to create authentication aliases for use by the MQ messaging provider activation specification.

**Activation specifications on z/OS**

Several additional features are available when using the MQ messaging provider activation specifications on the z/OS platform. These features are designed to take advantage of the unique scalability and workload management features of z/OS, and include a split process activation specification implementation to minimize queue manager side message contention, as well as z/OS workload management support for activation specifications.

See [Related topics](#) for more information on this functionality, as well as how to enable and configure it.

**Migrating message listener ports to activation specifications**

Users of earlier iterations of the MQ messaging provider used message listener ports to deliver messages from WebSphere MQ messaging provider destinations to MDBs running in WebSphere Application Server. When migrating to WebSphere Application Server V7, MQ messaging provider activation specifications can be used instead. Several tools have been provided to make the task of migrating to activation specifications easier.

When migrating from message listener ports, the first task is to create a new MQ messaging provider activation specification. This can be achieved by either creating an activation specification from scratch, or by using tooling that will create a WebSphere MQ messaging provider activation specification based on the settings of a particular message listener port configuration.

From the wsadmin command line tool, you can use the `migrateWMQMLP` command to perform the migration. You can see an example invocation of this command in the [introduction of admin commands](#) section.

Alternatively, you can migrate using the WebSphere Application Server admin console. To do this for an existing message listener port:

1. Login to the administrative console.
2. Navigate to **Servers => Server Types => WebSphere application servers** (Figure 5) and click on the relevant application server. On the right side, under the Communications heading, expand the **Messaging** tab and click the **Message listener service** link. Under Additional properties, click on the **Listener Ports** link.
3. Select the message listener port to be converted to an MQ messaging provider activation specification, and click the **Convert to activation specification** button (Figure 6).

**Figure 6. Convert to activation specification**

4. If desired, change the values for the administrative name and JNDI name for the new MQ activation specification. Also, choose the relevant scope at which the activation specification should be defined. Because activation specifications can be created at any scope, you probably won’t need to migrate all message listener port configurations, especially in a clustered environment. Take care to avoid JNDI name clashes if multiple message listener port configurations are migrated. Click **Next**.
5. View the summary information, and if everything is correct, click **Finish**.

Once the message listener port configuration has been migrated, the relevant MDBs should be reconfigured to use the new MQ messaging provider activation specification. When the message listener port is no longer referenced by any MDBs, it can safely be deleted, either via the administrative console or by wsadmin.

### Changes to connection factories

The MQ messaging provider in WebSphere Application Server V7 continues to provide full support for connection factories in much the same way as it has in prior releases. However, a number of enhancements have been added to make it easier to create and configure connection factories, in addition to support for certain features. This section provides an overview of these enhancements.

#### Connection factory creation wizard

The process of using the administrative console to create an MQ messaging provider connection factory is now more manageable in WebSphere Application Server V7 with the introduction of a connection factory creation wizard. The wizard provides an easy way of creating a basic connection factory configuration by prompting you for the most common configuration properties. You can then use the enhanced MQ messaging provider connection factory panels to tailor the new connection factory to your specific requirements.

Launching this wizard is done in much the same way as for **activation specifications**, except that you select the type of connection factory that you want to create.

#### New connection factory features

Full support for a number of important MQ messaging provider connection factory configuration parameters has been added in WebSphere Application Server V7. Probably the most significant of these is **better integration with the SSL support** in WebSphere Application Server, and full support for creating messaging provider connection factories that are based off of CCDTs. Messaging provider connection factories now also expose support for **message header and payload compression**.
Redesigned connection factory administrative panels

The structure and layout of the administrative console panels for MQ messaging provider connection factories has been changed significantly in WebSphere Application Server V7. This restructuring makes it easier for you to configure an MQ messaging provider connection factory by grouping related properties together in various sections over a number of different panels:

- The **General properties** panel (Figure 8) displays the most often used configuration properties, such as administrative information, connection configuration, security settings, and so on, and provides links to the other panels, listed under the Additional Properties heading.
- The **Advanced properties** panel is used to configure more specialized options, such as message header and payload compression attributes, prefixes for temporary destinations, and connection consumer settings.
- The **Broker properties** panel is only available if a unified or topic connection factory is created, and provides a chance to configure the various capabilities and queues that are used for pub/sub-based messaging.
- Create custom properties in the form of name-value pairs using the **Custom properties** panel. These custom properties can be used to configure connection factory settings that are not available via WebSphere Application Server V7 administrative tooling.
- If a non-CCDT-based MQ messaging provider connection factory is created, the **Client transport properties** panel can be used to add and configure channel exits, as well as provide access to more advanced SSL configuration properties.
- The **Connection and session pool** panels can be used to change the default settings of the pools that are used to manage the connection and session objects created from a particular connection factory. It is worth noting that there is a single connection pool per MQ messaging provider connection factory, and each connection in the pool has its own session pool. For example, default pool sizes of 10 mean that, at most, there will be ten connections each with a pool of ten sessions, giving 100 sessions in total.
- The **JAAS - J2C authentication data** panel can be used to create authentication aliases for use by the MQ messaging provider connection factory.
Better support for SSL

The Secure Sockets Layer (SSL) protocol provides secure communications between remote server processes or endpoints. SSL security can be used for establishing communications inbound from (and outbound to) an endpoint. To establish secure communications, a certificate and an SSL configuration must be specified for the endpoint.

In prior versions of WebSphere Application Server, SSL-based connections to WebSphere MQ relied upon JVM custom properties to be defined (mainly for client container-based environments), or the node’s default SSL configuration to specify the keystore and truststore to be used, and the SSL ciphersuite was specified on a per connection factory basis.

WebSphere Application Server V7 contains several key changes in the way SSL configuration data is specified:

- Instead of using the default node configuration for SSL, a new outbound endpoint security configuration has been added specifically for WebSphere MQ client connections (named WebSphere MQ Client), allowing for customization of the default SSL configuration for WebSphere MQ connections to be modified separately from the node’s default configuration.
- When defining an MQ messaging provider connection factory or activation specification, there are now two ways you can obtain the SSL configuration data:
  - **Centrally managed**: When the connection factory or activation specification is used in WebSphere Application Server V7, the new WebSphere MQ Client endpoint security
configuration for that server will be used when you select this option. In the case of using connection factories in the client container, the settings from the ssl.client.props file of that client will be used.

- **Specific configuration:** You can use any defined SSL configuration within the cell.
  - Within the new WebSphere MQ Client endpoint, it is possible to define dynamic outbound endpoint SSL configurations based upon the target host and port information.
  - The SSL ciphersuite is now obtained from the SSL configuration rather than the connection factory. This removes the need to manually enter the ciphersuite on each connection factory that was created, although it is still possible to override the SSL ciphersuite to be used on a per connection factory basis.

See [Related topics](#) for more information regarding the configuration of SSL with the WebSphere MQ messaging.

**Full support for the CCDT**

In WebSphere Application Server V7, there are now two ways of specifying the information needed by MQ messaging provider connection factories or activation specifications so they can connect to a WebSphere MQ queue manager. The first approach requires you to enter all the information manually. The second approach is to provide the MQ messaging provider resource with a uniform resource locator (URL) that points to a client channel definition table (CCDT).

A CCDT is a binary file that contains information about how to create a client connection channel to one or more queue managers. The file contains information such as the hostname, port, and name of the target queue manager, as well as more advanced configuration information like the SSL attributes that should be used. A CCDT can be generated by a WebSphere MQ queue manager, or with the standalone tool provided by [SupportPac MO72](#).

Creating MQ messaging provider resources using a CCDT provides benefits that include:

- **Client connection channel information is contained in a single place.** If any of the information changes, such as the host name of the machine on which the WebSphere MQ queue manager resides, only the CCDT needs to be updated, and all WebSphere MQ messaging provider resources that make use of the CCDT will pick up the change.

- **Since less information is required, there is a reduced chance of configuration errors.** When using a CCDT to enter connection information, all that is required are the CCDT URL and an optional queue manager name. If you configure an MQ messaging provider resource manually, much more information is required -- especially if you’re configuring SSL.

See [Related topics](#) for more information on configuring MQ messaging provider resources using a CCDT.

**Channel exits written in Java**

Channel exits are user code that is run at defined points in the life cycle of a WebSphere MQ channel. There are many potential uses for channel exits including auditing, security, compression,
conversion, and so on. The MQ messaging provider in WebSphere Application Server V7 provides full support for Java-based channel exits.

In prior versions of WebSphere Application Server, channel exits had to be configured using custom properties. In WebSphere Application Server V7, channel exits can be configured on both connection factories and activation specifications using either the admin console, admin commands, or by specifying them in a CCDT entry.

To make use of a channel exit, an MQ messaging provider connection factory or activation specification must have a transport type of "client" or "bindings then client." This is because channel exit programs can only be used when connecting to a WebSphere MQ queue manager or queue sharing group using a client channel-based connection. If "bindings then client" transport mode is chosen, channel exits will only be driven if the bindings mode connection fails.

The MQ messaging provider supports three different type of channel exit:

- A **security exit** is invoked during channel start up and, as the name suggests, is typically used for authorization purposes. However, security exits are useful for running any other code that only needs to be run when the channel starts up.
- **Send exits** and **receive exits** are invoked immediately before a transmission is sent over a channel and immediately after a transmission is received over a channel, respectively. Possible uses for send and receive exits include compression or logging.

### Automatic selection of transport types

When an enterprise application uses the MQ messaging provider to connect to a queue manager, there are two primary ways in which this connection is made. One option is to use a TCP/IP-based network connection; this is referred to as a **client mode** connection. The other option is to use cross-memory inter-process communications, often referred to as a **bindings mode** connection. Although the choice of connection mechanism is transparent to the enterprise application, this information, called the **transport mode**, forms part of the configuration of MQ messaging provider connection factories and activation specifications.

In configurations where the MQ queue manager is running on the same node as WebSphere Application Server, using bindings mode offers a performance advantage. When the queue manager and application server are hosted by different machines connected by a network, then a client mode connection is required. When using the new "bindings then client" mode, a bindings mode connection to the queue manager will be tried first, and if this connection is not possible, then a client mode connection will be established.

A transport mode of "bindings then client" can be selected when creating or modifying a WebSphere MQ messaging provider activation specification or connection factory.

### Channel compression

When enabling the new channel compression support in WebSphere Application Server V7, the contents of messages are compressed as they are transferred over the network. This compression
is applied to the data transferred across the link between enterprise applications, running in the application server, and the queue manager to which they are connected.

If channel compression is used, the manipulation of message data is done transparently from an application point of view; the message data is automatically uncompressed before being queued by WebSphere MQ. This makes use of channel compression equally transparent to other WebSphere MQ applications connecting to the queue manager. Figure 9 shows the key points at which message data is compressed and de-compressed.

**Figure 9. Channel compression**

The use of channel compression can help increase the speed at which messages are transferred between an application server and a queue manager. This is because compressing the data sent or received in a message reduces the amount of network bandwidth required to transfer the message.

In some circumstances, using channel compression can reduce the amount of processor time spent encrypting data for SSL-based connections. This is because the amount of time taken to encrypt data is, typically, dependent on the amount of data being encrypted and not the actual content of the data. Therefore, if your data compresses well, you might find that the amount of processor time taken to compress then encrypt the data is significantly less than the time it would have taken to encrypt all the uncompressed data.

WebSphere Application Server V7 supports two types of compression:

- The run length encoding (RLE) compression algorithm requires very little processor time and works well for data that has lots of the same byte value repeated in consecutive locations. It is not effective for data that does not have this level of repetition.
- The ZLIB compression algorithm requires more processor time but works well when applied to more varied data.

Channel compression can be enabled either through the WebSphere Application Server admin console, or via the wsadmin command line interface. To enable channel compression using the admin console (Figure 10):

1. Login to the administrative console.
2. Navigate to the **Resources** section, expand the **JMS** tab, and click on either **Activation specifications** or the appropriate type of connection factory link.
3. Click on the connection factory or activation specification on which to enable channel compression, and select the **Advanced properties** link on the right side.
4. At the top of the page, there is a Message compression section. To enable header compression, select the **Compress message headers** box, as indicated in the figure by the red arrow. To compress the message payload, select the desired compression algorithm from the drop down box, as indicated below by the blue arrow.

**Figure 10. Enable channel compression**

Introduction of admin commands

Prior to V7, it was possible to configure MQ messaging provider resources in WebSphere Application Server either through the admin console or by directly manipulating the WebSphere Application Server configuration model using the `wsadmin` tool.

Directly interacting with WebSphere Application Server configuration model via `wsadmin` has its drawbacks and can be error prone. A new set of administrative commands has been introduced in WebSphere Application Server V7 to enable the creation, modification, deletion, and display of MQ messaging provider resources, plus related functionality, such as migrating message listener ports to MQ messaging provider activation specifications.

The new commands take in several parameters and perform the appropriate action on the underlying configuration documents. Making use of these commands provides several key benefits over interacting with the configuration model directly:

- The commands provide input validation, helping to ensure that any changes to the configuration model are correct. This prevents startup errors caused by a corrupt configuration model.
- The commands are documented and provide command line help information to aid users.

All the MQ messaging provider administrative commands are part of the `WMQAdminCommands` command group, and can be run using either Jython or JACL. Get a summary of all the different commands by entering the following Jython command into a `wsadmin` prompt:

```python
wsadmin>print AdminTask.help('WMQAdminCommands')
```

The `WMQAdminCommands` command group is made up of six sets of commands, described in the following sections. When creating an MQ messaging provider resource, in general, the administrative name and JNDI name must be unique at the scope it is created, or an error message will result.

Activation specification commands

Five commands are provided for manipulating WebSphere MQ messaging provider activation specifications:
• **createWMQActivationSpecification**: Create a new messaging provider activation specification at a particular scope. For example, to create a simple CCDT-based WebSphere MQ messaging provider activation specification at node scope:

```java
wsadmin>node = AdminConfig.list('Node')
wsadmin>AdminTask.createWMQActivationSpec(node, 
[-name MyActivationSpec 
-jndiName 'jms/MyActivationSpec' -destinationJndiName 'jms/MyQueue' 
-destinationType javax.jms.Queue -ccdtUrl 'file://ccdt.tab' -ccdtQmgrName QM12])
```

• **listWMQActivationSpecs**: List all currently configured MQ messaging provider activation specifications defined at a particular scope:

```java
wsadmin>AdminTask.listWMQActivationSpecs(node)
['MyActivationSpec(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CActivationSpec_1219672912125)
MyOtherActivationSpec(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CActivationSpec_1219673212812)'
```

• **showWMQActivationSpec**: Display each activation specification individually:

```java
wsadmin>AdminTask.showWMQActivationSpec('MyActivationSpec(' 
cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CActivationSpec 
_1219672912125)')
```

• **modifyWMQActivationSpec**: Alter the configuration of an existing MQ messaging provider activation specification. (It is not possible to change an existing CCDT-based activation specification to a non-CCDT-based one.) For example, to modify a messaging provider activation specification so that it refers to a different destination in JNDI:

```java
wsadmin>AdminTask.modifyWMQActivationSpec('MyActivationSpec(' 
cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CActivationSpec 
_1219672912125)', 
[-destinationJndiName 'jms/AnotherWMQQueue'])
```

• **deleteWMQActivationSpec**: Delete an MQ messaging provider activation specification. For example, to delete the activation specification that was created earlier:

```java
wsadmin>AdminTask.deleteWMQActivationSpec('MyActivationSpec(' 
cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CActivationSpec 
_1219672912125)')
```

## Connection factory commands

Another set of five commands is provided for manipulating MQ messaging provider connection factories. These commands will work for all three types of connection factories, unified, queue, and topic.

• **createWMQConnectionFactory**: Create an MQ messaging provider connection factory at a particular scope. The type parameter specifies whether a unified, queue, or topic based connection factory is to be created and takes a value of CF, QCF, or TCF respectively. For example, to create a unified connection factory that is configured with send exits, receive exits, and then default values for everything else:

```java
wsadmin>AdminTask.createWMQConnectionFactory(node, 
[-name MyCF 
-jndiName 'jms/MyCF' -type CF -rcvExit com.example.ReceiveExit -rcvExitInitData 'debug=true' -sendExit com.example.SendExit -sendExitInitData 'compression=normal'])
```

• **modifyWMQConnectionFactory**: Modify an existing MQ messaging provider connection factory. This command cannot be used to change the type of connection factory. For example, to remove the send and receive channel exit configuration from the connection factory created above:
wsadmin>AdminTask.modifyWMQConnectionFactory('MyCF(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#MQConnectionFactory_1219674863640)', ['-rcvExit '' -rcvExitInitData '' -sendExit '' -sendExitInitData '' '])

- **listWMQConnectionFactory, showWMQConnectionFactory, deleteWMQConnectionFactory**: List all MQ messaging provider connection factories at a particular scope, show an individual MQ messaging provider connection factory, or delete an MQ messaging provider connection factory, respectively. For example to validate that the channel exit parameters are no longer set on MyCF:

wsadmin>AdminTask.showWMQConnectionFactory('MyCF(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#MQConnectionFactory_1219674863640)')

### Destination commands

Two sets commands are provided for creating and manipulating MQ messaging provider queues and topics. The **createWMQTopic** and **createWMQQueue** commands can be used to create a messaging provider queue or topic at the relevant scope. For example, to create an MQ messaging provider queue at the server scope:

wsadmin>server = AdminConfig.list('Server')
wsadmin>AdminTask.createWMQQueue(server, ['-name MyQueue -jndiName 'jms/MyQueue' -queueName q1.in -useNativeEncoding false -integerEncoding Reversed -decimalEncoding Reversed -floatingPointEncoding IEEEReversed'])

Similarly, to create a server scoped WebSphere MQ messaging provider topic:

wsadmin>AdminTask.createWMQTopic(server, ['-name MyTopic -jndiName 'jms/MyTopic' -topicName sport/football -useNativeEncoding false -integerEncoding Reversed -decimalEncoding Reversed -floatingPointEncoding IEEEReversed'])

- To list all WebSphere MQ messaging provider topics or queues at a particular scope, use the **listWMQTopics** and **listWMQQueues** commands, providing the name of the required scope as a parameter. To view the configuration of a particular MQ messaging provider queue or topic use the **showWMQTopic** or **showWMQQueue** commands, passing in the object name to be displayed. For example, to display the topic created earlier enter:

wsadmin>AdminTask.showWMQTopic('MyTopic(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#MQTopic_1219676983984)')

- The **modifyWMQQueue** and **modifyWMQTopic** commands can be used to modify the properties of a particular MQ messaging provider topic or queue. For example, this modifyWMQQueue command can be used to set useNativeEncoding back to the default of true:

wsadmin>AdminTask.modifyWMQQueue('MyQueue(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#MQQueue_1219676256312)', ['"-useNativeEncoding true"'])

Then, to validate that the change has occurred correctly, you can run the showWMQQueue command:

wsadmin>AdminTask.showWMQQueue('MyQueue(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#MQQueue_1219676256312)')
The **manageWMQ** command

The **manageWMQ** command has three uses:

- It can provide information about the version of the MQ messaging provider instance that can be used for debug purposes using the query option.
- It can set native path information using the nativePath option, which is needed if bindings mode connections to a queue manager will be used.
- It can enable the control region adjust process for inbound message delivery on the z/OS platform with the enableInbound option. If either the nativePath or enableInbound options are used, then the relevant application server processes must be recycled in order for the changes to take effect.

When invoking the manageWMQ command, the object name of an installed resource adapter (server, node, or cell scope) needs to be provided. This information can be obtained by looking for the relevant J2CResourceAdapter object. This example shows how to find these objects using wildcards:

```
wsadmin>AdminConfig.list('J2CResourceAdapter', '*WebSphere MQ*')
```

Here is an example of how to obtain version information about the installed MQ messaging provider at the node scope returned from the previous example:

```
wsadmin>AdminTask.manageWMQ("WebSphere MQ Resource Adapter(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#J2CResourceAdapter_1216206449734)", ['-query'])
```

When specifying native path information using the nativePath option, path information at a server scope will override path information at both the node and cell scopes. Similarly, setting native path information at the node scope overrides configuration specified at the cell scope. The path provided when using this option must specify the full path of the directory that contains the native libraries. For example, to specify native path information at the cell scope:

```
wsadmin>AdminTask.manageWMQ("WebSphere MQ Resource Adapter(cells/L3A3316Node04Cell|resources.xml#J2CResourceAdapter_1216206454984)", ['-nativePath','c:/Program Files/IBM/WebSphere MQ//Java//lib//'])
```

Below is an example of using the enableInbound option. When using this option, the J2CResourceAdapter object passed in must be at the server scope or an error will result:

```
wsadmin>AdminTask.manageWMQ("WebSphere MQ Resource Adapter(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#J2CResourceAdapter_1216206454812)", ['-enableInbound'])
```

The **migrateWMQMLP** command

The **migrateWMQMLP** command can be used to migrate a message listener port that uses MQ messaging provider resources to an MQ messaging provider activation specification. For example, to find and select a particular message listener port and then convert it to an MQ messaging provider activation specification at the server scope:
wsadmin> lp1 = AdminConfig.list('ListenerPort')
wsadmin> AdminTask.migrateWMQMLP(lp1, ['-asName as1 -asJNDIName jms/as1
-asScope server'])

Summary

This article provided a high level overview of the new functions that are available in the WebSphere MQ messaging provider in WebSphere Application Server V7, along with examples that described how to make use of these functions.
Related topics

- WebSphere Application Server Information Center
- WebSphere MQ Information Center
- JMS 1.1. specification and documentation
- WebSphere Message Broker product information
- Run Length Encoding (RLE)
- ZLIB compression library

© Copyright IBM Corporation 2009
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)