◆ ◆ ◆ CHAPTER 16

Administering the Java Message Service (JMS)

The Java Message Service (JMS) API is a messaging standard that allows Java EE applications and components, including message-driven beans (MDBs), to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

GlassFish Server supports JMS messaging by communicating with a *JMS provider* through a Java EE Connector resource adapter. By default, GlassFish Server provides JMS messaging through its built-in *jmsra* resource adapter communicating with Open Message Queue, which is included with GlassFish Server. This combination, known as the *JMS Service*, is tightly integrated with GlassFish Server, providing a rich set of `asadmin` subcommands and Administration Console pages to simplify JMS messaging administration tasks.

GlassFish Server also supports the Generic Resource Adapter for JMS (GenericJMSRA), available as an Add-On in the Administration Console's Update Tool, for use as a resource adapter to connect to other JMS providers. The last section in this chapter, "Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers" on page 304, describes the GenericJMSRA and provides instructions for using it to make other supported JMS providers available to GlassFish Server.

The following topics are addressed here:

- "About the JMS Service" on page 290
- "Updating the JMS Service Configuration" on page 291
- "Administering JMS Hosts" on page 293
- "Administering JMS Connection Factories and Destinations" on page 297
- "Administering JMS Physical Destinations" on page 301
- "Troubleshooting the JMS Service" on page 304
- "Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers" on page 304

Instructions for accomplishing the task in this chapter by using the Administration Console are contained in the Administration Console online help.

About the JMS Service

To support JMS messaging, the JMS Service provides the following administrative objects:

JMS Service Configuration
The JMS service configuration is part of the overall configuration for a GlassFish standalone instance or cluster. It specifies how the JMS Service is to create and maintain connections with JMS Hosts.

JMS Hosts
JMS hosts are the message servers that host destinations, store messages, and interact with applications to send and receive messages across connections. In Message Queue, JMS hosts are called *brokers*.

The JMS service supports these types of JMS hosts:

- **Embedded** type, in which the JMS host runs in the same JVM as the GlassFish instance; its configuration and lifecycle are managed by the JMS service
- **Local** type, in which the JMS host runs separately on the same host as the GlassFish instance; its configuration and lifecycle are managed by the JMS service
- **Remote** type, in which the JMS host represents a Message Queue broker or broker cluster that is external to the JMS service; its operation is managed using Message Queue administrative tools

For more information about JMS host types, see "About JMS Host Types" on page 293.

JMS Connection Factory Resources
JMS connection factory resources house the information that applications use to connect to a JMS provider. For each JMS connection factory, the JMS service automatically maintains a GlassFish connector resource and a GlassFish connector connection pool in order to support connection pooling and failover.

JMS Destination Resources
JMS destination resources house the information that applications use to specify the target destination of messages they produce and the source destination of messages they consume. For each JMS destination resource, the JMS service automatically maintains a GlassFish administered object.

JMS Physical Destinations
JMS physical destinations provide a means to create and manage JMS destinations administratively instead of having them created dynamically when needed by an application. While dynamic creation of destinations is often sufficient during application development, administratively created physical destinations are more suitable for production environments.

JMS Service High Availability

Just as GlassFish Server supports clusters of instances to provide high availability, Message Queue supports clusters of brokers to provide service availability or service and data availability, depending on the type of broker cluster, as described in Chapter 4, "Broker Clusters," in *Oracle GlassFish Message Queue 4.5 Technical Overview*.

The JMS service takes advantage of this Message Queue capability and automatically creates and manages a Message Queue broker cluster when a GlassFish cluster's configuration specifies Embedded or Local type JMS hosts. Additionally, both GlassFish clusters and standalone instances can use Message Queue broker clusters as Remote type JMS hosts.

For information about how the JMS service supports GlassFish clusters and Message Queue broker clusters, see Chapter 11, "Java Message Service Load Balancing and Failover," in *Oracle GlassFish Server 3.1 High Availability Administration Guide*.

Updating the JMS Service Configuration

Because the JMS service configuration is part of the overall configuration for a standalone instance or cluster, it is created when the standalone instance or cluster is created. You can then update the JMS service configuration by using the Java Message Service page for the configuration in the Administration Console, or by using a `set` subcommand of the following form:

`set configs.config.`*`config-name`*`.jms-service.`*`attribute-name`*`=`*`attribute-value`*

The attributes you can set are:

`type`
: The JMS host type the service is to use. Available choices are `EMBEDDED`, `LOCAL` and `REMOTE`. See "About JMS Host Types" on page 293 for more information.

`init-timeout-in-seconds`
: The number of seconds GlassFish Server waits for the JMS service to start before aborting the startup.

`start-args`
: A list of arguments the JMS service passes to Embedded and Local type JMS hosts on startup. Permissible arguments are the options supported by the Message Queue `imqbrokerd` command, as described in "Broker Utility" in *Oracle GlassFish Message Queue 4.5 Administration Guide*.

`default-jms-host`
: The name of the default JMS host.

`reconnect-enabled`
When set to `true`, the JMS service attempts to reconnect to a JMS host (or one of the JMS hosts in the AddressList) when a connection is lost.

`reconnect-attempts`
The number of attempts to connect (or reconnect) for each JMS host in the AddressList before the JMS service tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the JMS service attempts to connect to the first address until it succeeds).

`reconnect-interval-in-seconds`
The number of seconds between reconnect attempts. This interval applies for attempts on each JMS host in the AddressList and for successive addresses in the list. If it is too short, this time interval does not give a JMS host time to recover. If it is too long, the reconnect might represent an unacceptable delay.

`addresslist-behavior`
The order of connection attempts. Available choices are:

`random`
Select a JMS host from the AddressList randomly. If there are many clients attempting a connection using the same connection factory, specify `random` to prevent them from all being connected to the same JMS host.

`priority`
Always try to connect to the first JMS host in the AddressList and use another one only if the first one is not available.

`addresslist-iterations`
The number of times the JMS service iterates through the AddressList in an effort to establish (or reestablish) a connection. A value of -1 indicates that the number of attempts is unlimited.

`mq-scheme`
`mq-service`
The Message Queue address scheme name and connection service name to use for the AddressList if a non-default scheme or service is to be used. See "Connection Handling" in *Oracle GlassFish Message Queue 4.5 Administration Guide* for syntax information.

In addition to these attributes, you can specify any Message Queue broker property in the JMS service configuration by adding it by name to the Additional Properties table on the Java Message Service page for the configuration in the Administration Console, or by using a `set` subcommand of the following form:

`set configs.config.`*config-name*`.jms-service.property.`*broker-property-name*`=`*value*

If the broker property name includes dots, preface the dots with two backslashes (`\\`); for example, to set the `imq.system.max_count` property, specify `imq\\.system\\.max_count` in the `set` subcommand.

You can also set broker properties in the JMS host. If you set the same broker property in both the JMS service configuration and the JMS host, the value specified in the JMS host is used.

Note – After making changes to the JMS service configuration, GlassFish Server instances that use the configuration must be restarted in order for the changes to be propagated.

Administering JMS Hosts

A *JMS host* represents a Message Queue broker. JMS contains a *JMS hosts list* (the `AddressList` property) that contains all the JMS hosts that are used by GlassFish Server. The JMS hosts list is populated with the hosts and ports of the specified Message Queue brokers and is updated whenever a JMS host configuration changes. When you create JMS resources or deploy message driven beans, the resources or beans inherit the JMS hosts list.

The following topics are addressed here:

- "About JMS Host Types" on page 293
- "Configuring Embedded and Local JMS Hosts" on page 294
- "To Create a JMS Host" on page 295
- "To List JMS Hosts" on page 295
- "To Update a JMS Host" on page 296
- "To Delete a JMS Host" on page 297

For information about administering JMS hosts that are servicing GlassFish clusters, see "Configuring GlassFish Server Clusters to Use Message Queue Broker Clusters" in *Oracle GlassFish Server 3.1 High Availability Administration Guide*.

About JMS Host Types

The JMS service uses Message Queue (MQ) brokers as JMS hosts, integrating them in three ways:

Embedded Type

When the JMS service configuration's `type` attribute is `EMBEDDED`, the MQ broker is co-located in the same JVM as the GlassFish server instance it services. The JMS service starts it in-process and manages its configuration and lifecycle.

For this type, the JMS service uses lazy initialization to start the broker when the first JMS operation is requested instead of immediately when the GlassFish instance is started. Additionally, if the GlassFish instance is a standalone instance (not a clustered instance), JMS operations use a Message Queue feature called *direct mode* to bypass the networking stack, leading to performance optimization.

Local Type

When the JMS service configuration's `type` attribute is `LOCAL`, the JMS service starts the MQ broker specified in the configuration as the default JMS host in a separate process on the same host as the GlassFish server instance. The JMS service manages its configuration and lifecycle.

For this type, the JMS service provides the MQ broker an additional port to start the RMI registry. This port number is equal to the broker's JMS port plus 100. For example, if the JMS port number is 37676, then the additional port's number will be 37776. Additionally, the `start-args` property of the JMS service configuration can be used to specify MQ broker startup options.

Remote Type

When the JMS service configuration's `type` attribute is `REMOTE`, the JMS service uses the information defined by the default JMS host to communicate with an MQ broker or broker cluster that has been configured and started using Message Queue tools, as described in the *Oracle GlassFish Message Queue 4.5 Administration Guide*. Ongoing administration and tuning of the broker or broker cluster are also performed using Message Queue tools.

Configuring Embedded and Local JMS Hosts

Because the JMS service, not Message Queue, manages Embedded and Local JMS hosts automatically, you should avoid using Message Queue utilities to configure them. Instead, specify broker properties in the JMS service configuration, as described in "Updating the JMS Service Configuration" on page 291.

Should the need to use Message Queue utilities arise, you must use the `-varhome` option when running certain Message Queue utilities to specify the `IMQ_VARHOME` location of the Embedded or Local JMS host. This location depends on which GlassFish instance the JMS host is servicing:

- For `server`, the Domain Administration Server (DAS), the `IMQ_VARHOME` location is:

  domain-root-dir`/`*domain-name*`/imq`
- For any other GlassFish instance, the `IMQ_VARHOME` location is:

  as-install`/nodes/`*node-name*`/`*instance-name*`/imq`

For example, the broker log file for an Embedded or Local JMS host servicing the DAS is available at *domain-root-dir*`/`*domain-name*`/imq/instances/imqbroker/log/log.txt`, and the broker log file for an Embedded or Local JMS host servicing any other GlassFish instance is available at *as-install*`/nodes/`*node-name*`/`*instance-name*`/imq/instances/`*mq-instance-name*`/log/log.txt`.

▼ To Create a JMS Host

The default JMS service configuration includes a JMS host, `default_JMS_host`. For most situations, this host is sufficient, so replacing it or creating additional JMS hosts is not often necessary and is a task for advanced users. Use the `create-jms-host` subcommand in remote `asadmin` mode to create an additional JMS host.

Note – In addition to specifying the normal `create-jms-host` subcommand options when creating a JMS host, you can specify Message Queue broker properties by using the `--property` option:

`--property` *prop1Name=prop1Value*:*prop2Name=prop2Value*:...

If a broker property name includes dots, preface the dots with two backslashes (`\\`); for example, to include the `imq.system.max_count` property, specify `imq\\.system\\.max_count` in the `--property` option.

You can also set broker properties in the JMS service configuration. If you set the same broker property in both the JMS host and the JMS service configuration, the value specified in the JMS host is used.

---

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **Create the JMS host by using the `create-jms-host(1)` subcommand.**

Example 16–1 Creating a JMS Host

This example creates a JMS host named `MyNewHost`.

```
asadmin> create-jms-host --mqhost pigeon --mqport 7677
--mquser admin --mqpassword admin MyNewHost
Jms Host MyNewHost created.
Command create-jms-host executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help create-jms-host` at the command line.

▼ To List JMS Hosts

Use the `list-jms-hosts` subcommand in remote `asadmin` mode to list the existing JMS hosts.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2 **List the JMS hosts by using the `list-jms-hosts(1)` subcommand.**

Example 16–2 Listing JMS Hosts

The following subcommand lists the existing JMS hosts.

```
asadmin> list-jms-hosts
default_JMS_host
MyNewHost
Command list-jmsdest executed successfully
```

▼ To Update a JMS Host

Use the `set` subcommand in remote `asadmin` mode to update an existing JMS host.

1 **Ensure that the server is running.**

Remote `asadmin` subcommands require a running server.

2 **Use the `get(1)` subcommand to list the current attribute values of the desired JMS host:**

`asadmin> get configs.config.`*config-name*`.jms-service.jms-host.`*jms-host-name*`.*`

For information about JMS host attributes, see `create-jms-host(1)`.

3 **Use the `set(1)` subcommand to modify a JMS host attribute.**

Note – You can specify any Message Queue broker property in the JMS host configuration by setting the `property` attribute in the configuration using a `set` subcommand of the following form:

`set configs.config.`*config-name*`.jms-service.jms-host.`*jms-host-name*`.property.`*property-name*`=`*value*

If the broker property name includes dots, preface the dots with two backslashes (`\\`); for example, to set the `imq.system.max_count` property, specify `imq\\.system\\.max_count` in the `set` subcommand.

You can also set broker properties in the JMS service configuration. If you set the same broker property in both the JMS host and the JMS service configuration, the value specified in the JMS host is used.

Example 16–3 Updating a JMS Host

This example changes the value of the `host` attribute of the JMS host `default_JMS_Host`. By default this value is `localhost`.

```
asadmin> set configs.config.server-config.jms-service.jms-host.default_JMS_host.host=
"server1.middleware.example.com"
```

▼ To Delete a JMS Host

Use the `delete-jms-host` subcommand in remote `asadmin` mode to delete a JMS host from the JMS service. If you delete the only JMS host, the JMS service will not be able to start until you create a new JMS host.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **List the JMS hosts by using the `list-jms-hosts(1)` subcommand.**

3. **Delete a JMS host by using the `delete-jms-host(1)` subcommand.**

Example 16–4 Deleting a JMS Host

This example deletes a JMS host named `MyNewHost`.

```
asadmin> delete-jms-host MyNewHost
Command delete-jms-host executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help delete-jms-host` at the command line.

Administering JMS Connection Factories and Destinations

The JMS API uses two kinds of administered objects. *Connection factory objects* allow an application to create other JMS objects programmatically. *Destination objects* serve as repositories for messages. How these objects are created is specific to each implementation of JMS. In GlassFish Server, JMS is implemented by performing the following tasks:

- Creating a connection factory
- Creating a destination, which requires creating a physical destination and a destination resource that refers to the physical destination

JMS applications use the Java Naming and Directory Interface (JNDI) API to access the connection factory and destination resources. A JMS application normally uses at least one connection factory and at least one destination. By studying the application or consulting with the application developer, you can determine what resources must be created. The order in which the resources are created does not matter.

GlassFish Server provides the following types of connection factory objects:

- `QueueConnectionFactory` objects, used for point-to-point communication
- `TopicConnectionFactory` objects, used for publish-subscribe communication

- `ConnectionFactory` objects, which can be used for both point-to-point and publish-subscribe communications (recommended for new applications)

GlassFish Server provides the following types of destination objects:

- `Queue` objects, used for point-to-point communication
- `Topic` objects, used for publish-subscribe communication

The following topics are addressed here:

- "To Create a Connection Factory or Destination Resource" on page 298
- "To List JMS Resources" on page 299
- "To Delete a Connection Factory or Destination Resource" on page 300

The subcommands in this section can be used to administer both the connection factory resources and the destination resources. For information on JMS service support of connection pooling and failover, see "Connection Pooling and Failover" in *Oracle GlassFish Server 3.1 High Availability Administration Guide*. For instructions on administering physical destinations, see "Administering JMS Physical Destinations" on page 301.

▼ To Create a Connection Factory or Destination Resource

For each JMS connection factory that you create, GlassFish Server creates a connector connection pool and connector resource. For each JMS destination that you create, GlassFish Server creates a connector admin object resource. If you delete a JMS resource, GlassFish Server automatically deletes the connector resources.

Use the `create-jms-resource` command in remote `asadmin` mode to create a JMS connection factory resource or a destination resource.

Tip – To specify the `addresslist` property (in the format `host:mqport,host2:mqport,host3:mqport`) for the `asadmin create-jms-resource` command, escape the : by using `\\`. For example, `host1\\:mqport,host2\\:mqport,host3\\:mpqport`. For more information about using escape characters, see the `asadmin`(1M) concepts page.

To update a JMS connection factory, use the `set` subcommand for the underlying connector connection pool, See "To Update a Connector Connection Pool" on page 247.

To update a destination, use the `set` subcommand for the admin object resource. See "To Update an Administered Object" on page 261.

1 Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. **Create a JMS resource by using the `create-jms-resource(1)` command.**

   Information about the properties for the subcommand is included in this help page.

3. **(Optional) If needed, restart the server.**

   Some properties require server restart. See "Configuration Changes That Require Server Restart" on page 35. If your server needs to be restarted, see "To Restart a Domain" on page 89.

Example 16–5 Creating a JMS Connection Factory

This example creates a connection factory resource of type `javax.jms.ConnectionFactory` whose JNDI name is `jms/DurableConnectionFactory`. The `ClientId` property sets a client ID on the connection factory so that it can be used for durable subscriptions. The JNDI name for a JMS resource customarily includes the `jms/` naming subcontext.

```
asadmin> create-jms-resource --restype javax.jms.ConnectionFactory
--description "connection factory for durable subscriptions"
--property ClientId=MyID jms/DurableConnectionFactory
Command create-jms-resource executed successfully.
```

Example 16–6 Creating a JMS Destination

This example creates a destination resource whose JNDI name is `jms/MyQueue`.

```
asadmin> create-jms-resource --restype javax.jms.Queue
--property Name=PhysicalQueue jms/MyQueue
Command create-jms-resource executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help create-jms-resource` at the command line.

▼ To List JMS Resources

Use the `list-jms-resources` subcommand in remote `asadmin` mode to list the existing connection factory and destination resources.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **List the existing JMS resources by using the `list-jms-resources(1)` subcommand.**

Example 16–7 Listing All JMS Resources

This example lists all the existing JMS connection factory and destination resources.

```
asadmin> list-jms-resources
jms/Queue
jms/ConnectionFactory
jms/DurableConnectionFactory
jms/Topic
Command list-jms-resources executed successfully
```

Example 16–8 Listing a JMS Resources of a Specific Type

This example lists the resources for the resource type `javax`.

```
asadmin> list-jms-resources --restype javax.jms.TopicConnectionFactory
jms/DurableTopicConnectionFactory
jms/TopicConnectionFactory
Command list-jms-resources executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help list-jms-resources` at the command line.

▼ To Delete a Connection Factory or Destination Resource

Use the `delete-jms-resource` subcommand in remote `asadmin` mode to remove the specified connection factory or destination resource.

Before You Begin Ensure that you remove all references to the specified JMS resource before running this subcommand.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **List the existing JMS resources by using the `list-jms-resources(1)` subcommand.**

3. **Delete the JMS resource by using the `delete-jms-resource(1)` subcommand.**

Example 16–9 Deleting a JMS Resource

This example deletes the `jms/Queue` resource.

```
asadmin> delete-jms-resource jms/Queue
Command delete-jms-resource executed successfully
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help delete-jms-resource` at the command line.

Administering JMS Physical Destinations

Messages are delivered for routing and delivery to consumers by using *physical destinations* in the JMS provider. A physical destination is identified and encapsulated by an administered object (such as a `Topic` or `Queue` destination resource) that an application component uses to specify the destination of messages it is producing and the source of messages it is consuming. For instructions on configuring a destination resource, see "To Create a Connection Factory or Destination Resource" on page 298.

If a message-driven bean is deployed and the physical destination it listens to does not exist, GlassFish Server automatically creates the physical destination and sets the value of the `maxNumActiveConsumers` property to `-1`. However, it is good practice to create the physical destination beforehand. The first time that an application accesses a destination resource, Message Queue automatically creates the physical destination specified by the Name property of the destination resource. This automatically created physical destination is temporary and expires after a period specified by a Message Queue configuration property, provided that there are no messages in it and no message producers or consumers connected to it.

The following topics are addressed here:

- "To Create a JMS Physical Destination" on page 301
- "To List JMS Physical Destinations" on page 302
- "To Purge Messages From a Physical Destination" on page 302
- "To Delete a JMS Physical Destination" on page 303

▼ To Create a JMS Physical Destination

For production purposes, always create physical destinations. During the development and testing phase, however, this step is not required. Use the `create-jmsdest` subcommand in remote `asadmin` mode to create a physical destination.

Because a physical destination is actually a Message Queue object rather than a server object, you use Message Queue broker commands to update properties. For information on Message Queue properties, see *Oracle GlassFish Message Queue 4.4.2 Administration Guide*.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **Create a JMS physical destination by using the `create-jmsdest(1)` subcommand.**

   Information about the properties for the subcommand is included in this help page.

3. **(Optional) If needed, restart the server.**

   Some properties require server restart. See "Configuration Changes That Require Server Restart" on page 35. If your server needs to be restarted, see "To Restart a Domain" on page 89.

Example 16–10 Creating a JMS Physical Destination

This example creates a queue named `PhysicalQueue`.

```
asadmin> create-jmsdest --desttype queue --property
User=public:Password=public PhysicalQueue
Command create-jmsdest executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help create-jmsdest` at the command line.

▼ To List JMS Physical Destinations

Use the `list-jmsdest` subcommand in remote `asadmin` mode to list the existing JMS physical destinations.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **List the existing JMS physical destinations by using the `list-jmsdest(1)` subcommand.**

Example 16–11 Listing JMS Physical Destinations

This example lists the physical destinations for the default server instance.

```
asadmin> list-jmsdest
PhysicalQueue queue {}
PhysicalTopic topic {}
Command list-jmsdest executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help list-jmsdest` at the command line.

▼ To Purge Messages From a Physical Destination

Use the `flush-jmsdest` subcommand in remote `asadmin` mode to purge the messages from a physical destination in the specified target's JMS service configuration.

1. **Ensure that the server is running.**

   Remote `asadmin` subcommands require a running server.

2. **Purge messages from the a JMS physical destination by using the `flush-jmsdest(1)` subcommand.**

3 **(Optional) If needed, restart the server.**

Some properties require server restart. See "Configuration Changes That Require Server Restart" on page 35. If your server needs to be restarted, see "To Restart a Domain" on page 89.

Example 16–12 Flushing Messages From a JMS Physical Destination

This example purges messages from the queue named `PhysicalQueue`.

```
asadmin> flush-jmsdest --desttype queue PhysicalQueue
Command flush-jmsdest executed successfully
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help flush-jmsdest` at the command line.

▼ To Delete a JMS Physical Destination

Use the `delete-jmsdest` subcommand in remote `asadmin` mode to remove the specified JMS physical destination.

1 **Ensure that the server is running.**

Remote `asadmin` subcommands require a running server.

2 **List the existing JMS physical destinations by using the `list-jmsdest(1)` subcommand.**

3 **Delete the physical resource by using the `delete-jmsdest(1)` subcommand.**

Example 16–13 Deleting a Physical Destination

This example deletes the queue named `PhysicalQueue`.

```
asadmin> delete-jmsdest --desttype queue PhysicalQueue
Command delete-jmsdest executed successfully
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help delete-jmsdest` at the command line.

Troubleshooting the JMS Service

When you start GlassFish instances, Message Queue brokers acting as Local JMS hosts are started, but brokers acting as Embedded JMS hosts are not started until they are needed (for example, when you create a JMS resource). Use the `jms-ping(1)` subcommand to force the Message Queue broker for an Embedded host to start. If the `jms-ping` subcommand is unable to contact the broker, an error message is displayed.

If you encounter problems, consider the following:

- View the GlassFish Server log file. For `server`, the Domain Administrations Server (DAS), the log is available at *domain-dir*`/logs/server.log`; for other GlassFish instances, the log is available at *as-install*`/nodes/`*node-name*`/`*instance-name*`/logs/server.log`.

  If the log file indicates that a Message Queue broker acting as a Remote JMS host did not respond to a message, stop the broker and then restart it.
- View the broker log. For a broker associated with the Domain Administration Server (DAS), the log is available at *domain-dir*`/imq/instances/imqbroker/log/log.txt`; for brokers associated with other GlassFish instances, the log is available at *as-install*`/nodes/`*node-name*`/`*instance-name*`/imq/instances/`*mq-instance-name*`/log/log.txt`.
- For Remote type JMS hosts, be sure to start Message Queue brokers first, then GlassFish Server instances.
- If all Message Queue brokers are down, it can take up to 30 minutes for GlassFish Server to go down or up when you are using the default values in JMS. You can change the default values for this timeout. For example:

  ```
  asadmin set domain1.jms-service.reconnect-interval-in-seconds=5
  ```

Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers

GlassFish Server supports the integration and use of Oracle WebLogic JMS and IBM WebSphere MQ JMS providers through the use of the Generic Resource Adapter for JMS (GenericJMSRA), which is available as an Add-On in the Administration Console's Update Tool. This Java EE connector 1.5 resource adapter can wrap the JMS client library of Oracle WebLogic JMS and IBM WebSphere MQ and make it available for use by GlassFish. The adapter is a `.rar` archive that can be deployed and configured using GlassFish Server administration tools.

The following topics are addressed here:

- "Configuring GenericJMSRA for Supported External JMS Providers" on page 305
- "Using GenericJMSRA with WebLogic JMS" on page 312
- "Using GenericJMSRA with IBM WebSphere MQ" on page 325