

# Configuring Java Message Service Failover, Load Balancing and Clustering

---

This chapter describes how to configure connection failover, load balancing and clustering of the Java Message Service (JMS) for use with GlassFish Server.

The following topics are addressed here:

- [“Connection Failover” on page 161](#)
- [“Load-Balanced Message Inflow” on page 162](#)
- [“Using Message Queue Broker Clusters with GlassFish Server” on page 163](#)

## Connection Failover

GlassFish Server supports JMS connection failover, providing several options that control how connection failures are handled.

Use the Administration Console's Java Message Service page to configure these options. To display this page, click the configuration for the GlassFish cluster or instance in the navigation pane, and then click the Java Message Service link on the Configuration page.

### Reconnect

Enables connection failover. When disabled, the Java Message Service does not attempt to reconnect if a connection fails.

### Reconnect Interval

Specifies the number of seconds between reconnection attempts. If it is too short, this time interval does not give a broker time to recover. If it is too long, the wait time might represent an unacceptable delay. The default value is 5 seconds.

### Reconnect Attempts

Specifies the number of attempts to connect (or reconnect) to a particular JMS host before trying another host in the JMS host list, also known as the *Address List*. The default value is 3. A value of -1 indicates that the number of reconnect attempts is unlimited, which effectively limits connection attempts to the *primary JMS host*. The Java Message Service selects the primary JMS host based on the JMS host type in the JMS service configuration:

- For Embedded and Local JMS host types, the primary JMS host for a GlassFish instance is the one automatically created and managed for the instance by the Java Message Service.
- For the Remote JMS host type, the primary JMS host for a GlassFish instance is randomly chosen from the JMS host list.

### Address List Behavior

Specifies whether connection attempts are made in the order of JMS hosts in the JMS hosts list (priority) or random order (random). If set to Priority, the Java Message Service tries to connect to the first JMS host specified in the JMS hosts list and uses another one only if the first one is not available. If set to Random, the Java Message Service selects the JMS host randomly from the JMS hosts list. If there are many clients attempting a connection using the same connection factory, use the Random setting to prevent them from all attempting to connect to the same address. The default for Embedded and Local JMS host types is Priority, and the default for the Remote JMS host type is Random.

### Address List Iterations

Specifies the number of times the Java Message Service iterates through the JMS hosts list in an effort to establish (or re-establish) a connection). A value of -1 indicates that the number of attempts is unlimited.

You can override these settings using JMS connection factory settings. For details, see “Administering JMS Connection Factories and Destinations” in *GlassFish Server Open Source Edition 3.1 Administration Guide*.

## Load-Balanced Message Inflow

You can configure `ActivationSpec` properties of the `jmsra` resource adapter in the `sun-ejb-jar.xml` file for a message-driven bean using `activation-config-property` elements. Whenever a message-driven bean (`EndPointFactory`) is deployed, the connector runtime engine finds these properties and configures them accordingly in the resource adapter. See “`activation-config-property`” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

GlassFish Server transparently enables messages to be delivered randomly to message-driven beans having the same `ClientID`. The `ClientID` is required for durable subscribers.

For non-durable subscribers in which the `ClientID` is not configured, all instances of a specific message-driven bean that subscribe to same topic are considered equal. When a message-driven bean is deployed to multiple instances of GlassFish Server, only one of the message-driven beans receives the message. If multiple distinct message-driven beans subscribe to same topic, one instance of each message-driven bean receives a copy of the message.

To support multiple consumers using the same queue, set the `maxNumActiveConsumers` property of the physical destination to a value greater than zero. If this property is set, Message Queue allows up to that number of message-driven beans to consume messages from same

queue. The message is delivered randomly to the message-driven beans. If `maxNumActiveConsumers` is set to `-1`, there is no limit to the number of consumers.

To ensure that a message-driven bean is first connected to a co-located JMS host, set the `addressList-behavior` to `priority`. This setting specifies that the first broker in the `AddressList` is selected first. This first broker is the local co-located Message Queue broker. If this broker is unavailable, connection attempts are made to brokers in the order in which they are listed in the `AddressList`. This setting is the default for GlassFish Server instances and clusters whose JMS service configuration specifies `Embedded` or `Local` JMS host type.

## Using Message Queue Broker Clusters with GlassFish Server

This section describes how the JMS service uses Message Queue broker clusters to support high-availability JMS messaging in GlassFish Server clusters. It also provides instructions for configuring the types of Message Queue broker clusters to support the types of JMS hosts that GlassFish Server clusters can use.

The following topics are addressed here:

- “About Message Queue Broker Clusters” on page 163
- “Configuring GlassFish Clusters to Use Message Queue Broker Clusters” on page 164
- “To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster With Master Broker” on page 166
- “To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster of Peer Brokers” on page 166
- “To Configure a GlassFish Cluster to Use a Local Enhanced Broker Cluster” on page 168
- “To Change the Master Broker in an Embedded or Local Broker Cluster” on page 167
- “To Migrate Between Types of Embedded or Local Conventional Broker Clusters” on page 167
- “To Configure a GlassFish Cluster to Use a Remote Broker Cluster” on page 168

## About Message Queue Broker Clusters

The following discussion provides a brief overview of Message Queue broker clusters. For complete information, see Chapter 4, “Broker Clusters,” in *Open Message Queue 4.5 Technical Overview*.

Message Queue supports two clustering models both of which provide a scalable message service, but with each providing a different level of message service availability:

- **Conventional broker clusters.** A conventional broker cluster provides for *service availability*. When a broker fails, clients connected to the failed broker reconnect to another broker in the cluster. However, messages and state information stored in the failed broker cannot be recovered until the failed broker is brought back online. The broker failure can therefore result in a significant delay and in messages being delivered out of order.

Message Queue supports two types of conventional cluster, based on where the cluster configuration change record is stored:

- **Conventional cluster with master broker.** In a conventional cluster with a master broker, one of the brokers, designated as the *master broker*, stores and maintains the cluster configuration change record. The other brokers in the cluster must communicate with the master broker to keep abreast of changes to the cluster configuration. This is the simplest broker cluster to configure, and is the type of broker cluster that GlassFish Server uses by default to support GlassFish clusters.
- **Conventional cluster of peer brokers.** In a conventional cluster of peer brokers, the cluster configuration change record is stored in a JDBC data store accessible to all the brokers. Thus, brokers can access cluster configuration information whether any other brokers in the cluster are running or not.
- **Enhanced broker clusters.** An enhanced broker cluster provides for *data availability* in addition to service availability. When a broker fails, another broker takes over the pending work of the failed broker. The failover broker has access to the failed broker's messages and state information. Clients connected to the failed broker reconnect to the failover broker. In an enhanced cluster, as compared to a conventional cluster, a broker failure rarely results in significant delays in message delivery and messages are always delivered in order.

By its very nature, an enhanced broker cluster is a cluster of peer brokers.

---

**Note** – Despite the message service availability offered by both conventional and enhanced broker clusters, they do not provide a guarantee against failure and the possibility that certain failures, for example in the middle of a transaction, could require that some operations be repeated. It is the responsibility of the messaging application (both producers and consumers) to handle and respond appropriately to failure notifications from the messaging service.

---

## Configuring GlassFish Clusters to Use Message Queue Broker Clusters

When a GlassFish Server cluster is created, the JMS service automatically configures a Message Queue conventional broker cluster with master broker for the cluster, provided that the JMS host type in the GlassFish Server cluster's configuration is Embedded or Local. The JMS service configures one Message Queue broker for each instance in the GlassFish Server cluster, and designates as master broker the broker associated with the first instance created in the cluster. In the case of Local JMS hosts, the JMS service configures each broker to run on the same host as the instance with which it is associated. In the case of Embedded JMS hosts, the each broker inherently runs on the same host as the instance with which it is associated because it runs in the same JVM as the instance.

The JMS service manages the lifecycle of Embedded and Local JMS hosts, and this management extends to the management of Message Queue broker clusters as Embedded and Local JMS hosts. For a GlassFish cluster whose configuration specifies Embedded or Local JMS host type, the JMS service:

- Creates and manages one Message Queue broker for each instance in the GlassFish cluster, using this broker as the primary JMS host for the instance.
- Maintains the JMS host list for each instance in the GlassFish cluster such that its primary JMS host appears first in its JMS host list.

The JMS service supports the following types of Message Queue broker clusters with GlassFish Server clusters, based on the JMS host type:

#### **Embedded**

- Conventional broker cluster with master broker (default)
- Conventional broker cluster of peer brokers

#### **Local**

- Conventional broker cluster with master broker (default)
- Conventional broker cluster of peer brokers
- Enhanced broker cluster

#### **Remote**

- Conventional broker cluster with master broker; brokers can differ in number from GlassFish instances and can be located on other hosts
- Conventional broker cluster of peer brokers; brokers can differ in number from GlassFish instances and can be located on other hosts
- Enhanced broker cluster; brokers can differ in number from GlassFish instances and can be located on other hosts



**Caution** – After configuring a broker cluster for a GlassFish cluster using Embedded or Local JMS hosts, you must follow special procedures to migrate to another type of broker cluster, as described in [“To Migrate Between Types of Embedded or Local Conventional Broker Clusters”](#) on page 167.

The following topics provide instructions for configuring broker clusters in all these contexts.

## ▼ To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster With Master Broker

Use the `configure-jms-cluster` subcommand in remote `asadmin` mode to configure a conventional broker cluster with master broker to service a GlassFish Server cluster that uses either Embedded or Local JMS hosts.

Note that this configuration, with Embedded brokers, is the default for GlassFish Server clusters.

**Before You Begin** Perform the following steps after you have created the GlassFish Server cluster, but before you have added instances to the cluster or started the cluster.

### 1 Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

### 2 Configure the GlassFish Server cluster to use a Message Queue conventional broker cluster with master broker by using the `configure-jms-cluster(1)` subcommand:

```
> asadmin configure-jms-cluster --clustertype=conventional
--configstoretype=masterbroker glassfish-cluster-name
```

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

## ▼ To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster of Peer Brokers

Use the `configure-jms-cluster` subcommand in remote `asadmin` mode to configure a conventional broker cluster of peer brokers to service a GlassFish Server cluster that uses Embedded or Local JMS hosts.

**Before You Begin** Perform the following steps after you have created the GlassFish Server cluster, but before you have added instances to the cluster or started the cluster.

### 1 Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

### 2 Create a password file with the entry `AS_ADMIN_JMSDBPASSWORD` specifying the password of the database user.

For information about password file entries, see the `asadmin(1M)` command.

- 3 Place a copy of, or a link to, the database's JDBC driver .jar file in the *as-install-parent/mq/lib/ext* directory on each host where a GlassFish Server cluster instance is to run.
- 4 Configure the GlassFish Server cluster to use a Message Queue conventional broker cluster with master broker by using the `configure-jms-cluster(1)` subcommand:
 

```
> asadmin --passwordfile password-file configure-jms-cluster --clustertype=conventional
--configstoretype=shareddb --dbvendor database-vendor-name --dbuser database-user-name
--dburl database-url --property list-of-database-specific-properties glassfish-cluster-name
```

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

## ▼ To Change the Master Broker in an Embedded or Local Broker Cluster

Use the `change-master-broker` subcommand in remote `asadmin` mode to change the master broker to a different broker in a conventional broker cluster with master broker serving a GlassFish Server cluster that uses Embedded or Local JMS hosts.

Follow this procedure, for example, before you remove from a GlassFish cluster the instance associated with the current master broker.

- 1 **Ensure that the server is running.**  
Remote `asadmin` subcommands require a running server.
- 2 **Change the master broker by using the `change-master-broker(1)` subcommand:**  

```
> asadmin change-master-broker glassfish-clustered-instance-name
```

**See Also** You can also view the full syntax and options of the subcommand by typing `asadmin help change-master-broker` at the command line.

## ▼ To Migrate Between Types of Embedded or Local Conventional Broker Clusters

- If the need arises to convert from a conventional broker cluster with master broker to a conventional broker cluster of peer brokers, or the reverse, follow the instructions in “Managing Conventional Clusters” in *Open Message Queue 4.5 Administration Guide*.

## ▼ To Configure a GlassFish Cluster to Use a Local Enhanced Broker Cluster

Use the `configure-jms-cluster` subcommand in remote `asadmin` mode to configure an enhanced broker cluster to service a GlassFish Server cluster that uses Local JMS hosts.

**Before You Begin** Perform the following steps after you have created the GlassFish Server cluster, but before you have added instances to the cluster or started the cluster.

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

Remote `asadmin` subcommands require a running server.

**2 Create a password file with the entry `AS_ADMIN_JMSDBPASSWORD` specifying the password of the database user.**

For information about password file entries, see the `asadmin(1M)` command.

**3 Place a copy of, or a link to, the database's JDBC driver `.jar` file in the `as-install-parent/mq/lib/ext` directory on each host where a GlassFish Server cluster instance is to run.**

**4 Configure the GlassFish Server cluster to use a Message Queue enhanced broker cluster by using the `configure-jms-cluster(1)` subcommand:**

```
> asadmin --passwordfile password-file configure-jms-cluster --clustertype=enhanced
--configstoretype=shareddb --messagestoretype=jdbc
--dbvendor database-vendor-name --dbuser database-user-name --dburl database-url
--property list-of-database-specific-properties glassfish-cluster-name
```

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

## ▼ To Configure a GlassFish Cluster to Use a Remote Broker Cluster

**Before You Begin** Perform the following steps after you have:

- Used Message Queue to create a broker cluster.
- Created the GlassFish Server cluster, but not yet created instances for the cluster.

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

The remote subcommands used in this procedure require a running server.



**2 Delete the default\_JMS\_host JMS host by using the delete-jms-host(1) subcommand:**

```
> asadmin delete-jms-host --target glassfish-cluster-name default_JMS_host
```

**3 Create a JMS host for each broker in the broker cluster by using the create-jms-host(1) subcommand.**

For each broker, use an asadmin create-jms-host of the form:

```
> asadmin create-jms-host --target glassfish-cluster-name --mqhost broker-host  
--mqport broker-port --mquser mq-user --mqpassword mq-user-password  
jms-host-name-for-broker
```

**4 Start the brokers in the cluster by using the Message Queue imqbrokerd command, as described in “Managing Broker Clusters” in *Open Message Queue 4.5 Administration Guide*.****5 Create instances in the GlassFish Server cluster, as described in “[To Create an Instance Centrally](#)” on page 80 and “[To Create an Instance Locally](#)” on page 90.**

.