11

Configuring Java Message Service High Availability

This chapter describes how to configure the high availability features of the Java Message Service (JMS). It covers how to configure Message Queue broker clusters and how to use them to provide connection failover and load balancing, as described in the following topics:

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 describes the different cluster and broker types that are supported and how to configure them.

The following topics are addressed here:

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 *Oracle GlassFish 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 JMS message order semantics not being preserved.

    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, messages owned by the failed broker are delivered by the failover broker as soon as it takes over, and JMS message order semantics are preserved.

    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 respond to JMS exceptions appropriately. For information about the kinds of exceptions that can occur and how to respond to them, see "Handling Exceptions When Failover Occurs" in *Oracle GlassFish Server Message Queue 4.5 Developer's Guide for Java Clients*.

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

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.

> ⚠️ **Caution** – Before using this procedure to reconfigure an existing broker cluster, you must follow the 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 192. Failing to perform these special procedures could lead to data loss or corruption and even render your setup unusable, depending on the JMS operations performed on the existing 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.

⚠️ **Caution** – Before using this procedure to reconfigure an existing broker cluster, you must follow the 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 192. Failing to perform these special procedures could lead to data loss or corruption and even render your setup unusable, depending on the JMS operations performed on the existing 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.

Before You Begin To avoid possible issues regarding the synchronization of configuration state among the brokers in the cluster, make sure that all the brokers in the cluster are running before performing this procedure.

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** *Oracle GlassFish 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.

> **Caution –** Before using this procedure to reconfigure an existing broker cluster, you must follow the 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 192. Failing to perform these special procedures could lead to data loss or corruption and even render your setup unusable, depending on the JMS operations performed on the existing 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** *Oracle GlassFish Message Queue 4.5 Administration Guide* **for instructions..**

5   **Create instances in the GlassFish Server cluster, as described in "To Create an Instance Centrally" on page 84 and "To Create an Instance Locally" on page 94.**

.

Connection Failover

The use of Message Queue broker clusters provides JMS connection failover, including 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.

The way in which connection failover operates depends on whether the broker cluster is configured to be conventional or enhanced:

■   In a conventional cluster, when a broker fails, clients may reconnect to any other broker in the cluster. The Reconnect field specifies whether reconnection should take place, and the Address List Behavior and Address List Iterations fields specify how the client chooses what broker to fail over to.

■   In an enhanced cluster, when a broker fails, another broker automatically takes over its messages and clients. Clients automatically fail over to the appropriate broker. The Reconnect, Address List Behavior and Address List Iterations fields are ignored.

For more information on connection failover, including how failover on conventional clusters differs from failover on enhanced clusters, see "Automatic Reconnection" in *Oracle GlassFish Server Message Queue 4.5 Administration Guide*.

Reconnect
Applies only to conventional clusters. Enables reconnection and 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. The host list is also known as the *Address List*. Hosts are chosen from the address list either in order or randomly, depending on the setting of Address List Behavior.

Address List Behavior
For conventional clusters, this field specifies how the Java Message Service selects which JMS host in the JMS hosts list to initially connect to, and if the broker fails, how the Java Message Service selects which JMS host in the JMS hosts list to fail over to.

For enhanced clusters, this field specifies how the Java Message Service selects which JMS host in the JMS hosts list to initially connect to.

When performing initial connection or, for conventional clusters only, when performing failover, then if this attribute is 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 this attribute is set to Random, the Java Message Service selects the JMS host randomly from the JMS hosts list. If that host is not available, another one is chosen randomly.

The default for Embedded and Local JMS host types is Priority, and the default for the Remote JMS host type is Random.

For Embedded and Local JMS host types, the Java Message Service ensures that the Message Queue broker servicing a clustered instance appears first in that instance's JMS host list.

Thus, having Priority as the default Address List Behavior ensures that an application deployed to a clustered instance will always try to create its initial connection to that instance's co-located broker.

If there are many clients attempting a connection using the same connection factory, use the Random setting to prevent them from all attempting to create their initial connection to the same JMS host.

Address List Iterations
For conventional clusters, this field specifies the number of times the Java Message Service iterates through the JMS hosts list in an effort to establish its initial connection. If the broker fails, this field specifies the number of times the Java Message Service iterates through the JMS hosts list in an effort to fail over to another broker.

For enhanced clusters, this field specifies the number of times the Java Message Service iterates through the JMS hosts list in an effort to establish its initial connection. If the broker fails, this field is not used when performing reconnection.

You can override these settings using JMS connection factory settings. For details, see "Administering JMS Connection Factories and Destinations" in *Oracle GlassFish Server 3.1 Administration Guide*.

Load-Balanced Delivery to MDBs

When a message-driven bean (MDB) application is deployed to a GlassFish cluster, incoming messages are delivered randomly to MDBs without regard to the cluster instances in which they are running.

If the MDB is configured to receive messages from a durable or non-durable subscription on a topic, then only one MDB instance across the whole GlassFish cluster will receive each message.

For more information about these features, see "About Shared Topic Subscriptions for Clustered Containers" in *Oracle GlassFish Server Message Queue 4.5 Administration Guide*.