◆ ◆ ◆ CHAPTER 10

Java Message Service Load Balancing and Failover

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

The following topics are addressed here:

Connection Pooling and Failover

GlassFish Server supports JMS connection pooling and failover. The Oracle GlassFish Server pools JMS connections automatically. When the Address List Behavior attribute is `random` (the default), GlassFish Server selects its primary broker randomly from the JMS host list. When failover occurs, MQ transparently transfers the load to another broker and maintains JMS semantics. The default value for the Address List Behavior attribute is `priority`, if the JMS type is of type LOCAL.

To specify whether the GlassFish Server tries to reconnect to the primary broker when the connection is lost, select the Reconnect checkbox. If enabled and the primary broker goes down, GlassFish Server tries to reconnect to another broker in the JMS Hosts list.

When Reconnect is enabled, also specify the following attributes:

- **Address List Behavior**: whether connection attempts are in the order of addresses in the JMS Hosts List (priority) or random order (random). If set to Priority, Java Message Service tries to connect to the first MQ broker specified in the JMS Hosts list and uses another one only if the first broker is not available. If set to Random, Java Message Service selects the MQ broker randomly from the JMS Hosts list. If there are many clients attempting a connection using the same connection factory, use this setting to prevent them from all attempting to connect to the same address.

- **Address List Iterations**: 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.
- **Reconnect Attempts**: the number of attempts to connect (or reconnect) for each address in the JMS hosts list before the client runtime tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds).
- **Reconnect Interval**: number of seconds between reconnect attempts. This applies for attempts on each address in the JMS hosts list and for successive addresses in the list. If it is too short, this time interval does not give a broker time to recover. If it is too long, the reconnect might represent an unacceptable delay.

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 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 *Oracle GlassFish Server 3.1 Application Deployment Guide*.

The 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 the 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 large value. If this property is set, the Oracle Message Queue software 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 local delivery is preferred, set `addresslist-behavior` to priority. This setting specifies that the first broker in the `AddressList` is selected first. This first broker is the local colocated Message Queue instance. 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 that belong to a cluster.

Using Message Queue Broker Clusters with GlassFish Server

This section describes how the JMS service uses Message Queue broker clusters to support high-availabilty 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

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 or a connection 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 or connection 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 information about the cluster configuration 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 information about the cluster's configuration. The other brokers in the cluster must communicate with the master broker to keep abreast of changes to the configuration.
  - **Conventional cluster of peer brokers.** In a conventional cluster of peer brokers, the information about the cluster's configuration is stored in a JDBC data store accessible to all the brokers. Thus, brokers can access the 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 or a connection 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 or connection 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.

For more background information about Message Queue broker clusters, see Chapter 4, "Broker Clusters," in *Oracle GlassFish Message Queue 4.5 Technical Overview*.

Configuring GlassFish Server 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. The JMS service maintains the JMS host addresses and the JMS host list such that each instance in the GlassFish Server cluster uses the Message Queue broker configured for it as its primary broker and has that broker listed 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

Any, but manual configuration is required.

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

▼ To Configure a Conventional Broker Cluster With Master Broker as an Embedded or Local JMS Host for a GlassFish Server Cluster

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

Note that this configuration 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 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 Conventional Broker Cluster of Peer Brokers as an Embedded or Local JMS Host for a GlassFish Server Cluster

Use the `configure-jms-cluster` subcommand in remote 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 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 Configure an Enhanced Broker Cluster as a Local JMS Host for a GlassFish Server Cluster

Use the `configure-jms-cluster` subcommand in remote 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 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 with master broker by using the `configure-jms-cluster`(1) subcommand:**

```
> asadmin --passwordfile password-file configure-jms-cluster --clustertype=enhanced
--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 a Broker Cluster Serving as an Embedded or Local Host

Use the `change-master-broker` subcommand in remote 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.

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

   Remote 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 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 Integrate a Broker Cluster as a Remote JMS Host for a GlassFish Server 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 78and "To Create an Instance Locally" on page 88.**

   .