## Load Balancing With the Apache `mod_jk` Module

A common load balancing configuration for GlassFish Server 3.1 is to use the Apache HTTPD server as the Web server front-end, and the Apache `mod_jk` module as the connector between the Web Server and GlassFish Server. See "Configuring GlassFish Server with Apache HTTPD Server and `mod_jk`" on page 126 for more information.

## High Availability Session Persistence

GlassFish Server provides high availability of HTTP requests and session data (both HTTP session data and stateful session bean data).

Java EE applications typically have significant amounts of session state data. A web shopping cart is the classic example of a session state. Also, an application can cache frequently-needed data in the session object. In fact, almost all applications with significant user interactions need to maintain session state. Both HTTP sessions and stateful session beans (SFSBs) have session state data.

Preserving session state across server failures can be important to end users. If the GlassFish Server instance hosting the user session experiences a failure, the session state can be recovered, and the session can continue without loss of information. High availability is implemented in GlassFish Server by means of *in-memory session replication* on GlassFish Server instances running in a cluster.

For more information about in-memory session replication in GlassFish Server, see "How GlassFish Server Provides High Availability" on page 20. For detailed instructions on configuring high availability session persistence, see Chapter 9, "Configuring High Availability Session Persistence and Failover."

## High Availability Java Message Service

GlassFish Server supports the Java Message Service (JMS) API and JMS messaging through its built-in *jmsra* resource adapter communicating with Open Message Queue as the *JMS provider*. This combination is often called the *JMS Service*.

The JMS service makes JMS messaging highly available as follows:

**Connection Failover**

If the primary JMS host (Message Queue broker) in use by a GlassFish instance fails, Message Queue transparently transfers that broker's load to another JMS host in the JMS host list, maintaining JMS messaging semantics.

For more information about JMS connection failover, see "Connection Failover" on page 161.

**Message Queue Broker Clusters**

By default, when a GlassFish cluster is created, the JMS service automatically configures a Message Queue broker cluster to provide JMS messaging services, with one clustered broker assigned to each cluster instance. This automatically created broker cluster is configurable to take advantage of the two types of broker clusters, conventional and enhanced, supported by Message Queue.

Additionally, Message Queue broker clusters created and managed using Message Queue itself can be used as external, or remote, JMS hosts. Using external broker clusters provides additional deployment options, such as deploying Message Queue brokers on different hosts from the GlassFish instances they service, or deploying different numbers of Message Queue brokers and GlassFish instances.

For more information about Message Queue clustering, see "Using Message Queue Broker Clusters with GlassFish Server" on page 163.

## RMI-IIOP Load Balancing and Failover

With RMI-IIOP load balancing, IIOP client requests are distributed to different server instances or name servers, which spreads the load evenly across the cluster, providing scalability. IIOP load balancing combined with EJB clustering and availability also provides EJB failover.

When a client performs a JNDI lookup for an object, the Naming Service essentially binds the request to a particular server instance. From then on, all lookup requests made from that client are sent to the same server instance, and thus all `EJBHome` objects will be hosted on the same target server. Any bean references obtained henceforth are also created on the same target host. This effectively provides load balancing, since all clients randomize the list of target servers when performing JNDI lookups. If the target server instance goes down, the lookup or EJB method invocation will failover to another server instance.

IIOP Load balancing and failover happens transparently. No special steps are needed during application deployment. If the GlassFish Server instance on which the application client is deployed participates in a cluster, the GlassFish Server finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

For more information on RMI-IIOP load balancing and failover, see Chapter 11, "RMI-IIOP Load Balancing and Failover."

### Recovering Message Queue

JMS service configurations, including JMS host configurations for Embedded and Local JMS hosts, are stored in the Domain and are recovered when the Domain Administration Server (DAS) is recovered.

Most private data, such as file-based messages stores, maintained by a Message Queue broker acting as an Embedded or Local JMS host is stored with the GlassFish instance the broker is servicing and is recovered when the instance is recovered. However, if the broker is a member of an enhanced broker cluster, the private data is stored in a highly available database, and must be backed up and restored according to the database vendor's instructions.

Configuration information and private data for Message Queue brokers and broker clusters acting as Remote JMS hosts are stored in the Message Queue `IMQ_VARHOME` directory. Back up and restore these items using Message Queue utilities, as described in the *Open Message Queue 4.5 Administration Guide*.

## More Information

For information about planning a high-availability deployment, including assessing hardware requirements, planning network configuration, and selecting a topology, see the *GlassFish Server Open Source Edition 3.1 Deployment Planning Guide*. This manual also provides a high-level introduction to concepts such as:

- GlassFish Server components such as node agents, domains, and clusters
- IIOP load balancing in a cluster
- Message queue failover

For more information about developing applications that take advantage of high availability features, see the *GlassFish Server Open Source Edition 3.1 Application Development Guide*.

For information on how to configure and tune applications and GlassFish Server for best performance with high availability, see the *GlassFish Server Open Source Edition 3.1 Performance Tuning Guide*, which discusses topics such as:

- Tuning persistence frequency and persistence scope
- Checkpointing stateful session beans
- Configuring the JDBC connection pool
- Session size
- Configuring load balancers for best performance