



GlassFish Server Open Source Edition 3.1 High Availability Administration Guide

Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065
U.S.A.

Part No: 821-2452-13
August 2011

Copyright © 2010, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	11
1 High Availability in GlassFish Server	17
Overview of High Availability	17
Load Balancing With the Apache mod_jk Module	17
High Availability Session Persistence	18
High Availability Java Message Service	18
RMI-IIOP Load Balancing and Failover	19
How GlassFish Server Provides High Availability	19
Storage for Session State Data	20
Highly Available Clusters	20
Recovering from Failures	21
Recovering the Domain Administration Server	22
Recovering GlassFish Server Instances	22
Recovering the HTTP Load Balancer and Web Server	22
Recovering Message Queue	22
More Information	24
2 Setting Up SSH for Centralized Administration	25
About SSH for Centralized Administration	26
Determining Whether to Use SSH for Centralized Administration	26
Obtaining SSH Software	27
Determining the SSH User	27
Setting Up Cygwin SSH on Windows	28
▼ To Download and Install Cygwin	28
▼ To Set the Path for Windows and for the Cygwin Shell	29
▼ To Set the Home Directory for the Cygwin SSH User	30

▼ To Configure and Start the Cygwin SSH Server Daemon sshd	30
Setting Up the MKS Toolkit on Windows	31
▼ To Install the MKS Toolkit	32
▼ To Set the Path for Windows and for the MKS Toolkit Shell	32
▼ To Set the Home Directory for the MKS Toolkit SSH User	32
▼ To Configure and Start the MKS Toolkit SSH Server Daemon sshd	34
Setting Up SSH on UNIX and Linux Systems	35
▼ To Set Up SSH on Oracle Solaris Systems	36
▼ To Set Up SSH on MacOS Systems	37
▼ To Set Up SSH on Linux systems	37
Testing the SSH Setup on a Host	38
▼ To Test the SSH Setup on a Host	38
Setting Up SSH User Authentication	39
▼ To Set Up Public Key Authentication Without Encryption	40
▼ To Set Up Encrypted Public Key Authentication	42
▼ To Set Up Password Authentication	45
Installing and Removing GlassFish Server Software on Multiple Hosts	46
▼ To Copy a GlassFish Server Installation to Multiple Hosts	47
▼ To Remove GlassFish Server Software From Multiple Hosts	48
3 Administering GlassFish Server Nodes	51
Types of GlassFish Server Nodes	51
Creating, Listing, Testing, and Deleting SSH Nodes	52
▼ To Create an SSH Node	52
▼ To List SSH Nodes in a Domain	53
▼ To Test if an SSH Node is Reachable	54
▼ To Delete an SSH Node	55
Creating, Listing, and Deleting CONFIG Nodes	56
▼ To Create a CONFIG Node	56
▼ To List CONFIG Nodes in a Domain	58
▼ To Delete a CONFIG Node	59
Updating and Changing the Type of a Node	60
▼ To Update an SSH Node	60
▼ To Update a CONFIG Node	61
▼ To Change the Type of a Node	62

4	Administering GlassFish Server Clusters	65
	About GlassFish Server Clusters	65
	Group Management Service	66
	GMS Configuration Settings	66
	Dotted Names for GMS Settings	68
	▼ To Preconfigure Nondefault GMS Configuration Settings	69
	▼ To Change GMS Settings After Cluster Creation	70
	▼ To Check the Health of Instances in a Cluster	71
	▼ To Validate That Multicast Transport Is Available for a Cluster	71
	Using the Multi-Homing Feature With GMS	73
	Creating, Listing, and Deleting Clusters	75
	▼ To Create a Cluster	75
	▼ To List All Clusters in a Domain	78
	▼ To Delete a Cluster	78
5	Administering GlassFish Server Instances	81
	Types of GlassFish Server Instances	81
	Administering GlassFish Server Instances Centrally	82
	▼ To Create an Instance Centrally	82
	▼ To List All Instances in a Domain	85
	▼ To Delete an Instance Centrally	86
	▼ To Start a Cluster	87
	▼ To Stop a Cluster	88
	▼ To Start an Individual Instance Centrally	89
	▼ To Stop an Individual Instance Centrally	90
	▼ To Restart an Individual Instance Centrally	91
	Administering GlassFish Server Instances Locally	92
	▼ To Create an Instance Locally	92
	▼ To Delete an Instance Locally	96
	▼ To Start an Individual Instance Locally	98
	▼ To Stop an Individual Instance Locally	99
	▼ To Restart an Individual Instance Locally	100
	Resynchronizing GlassFish Server Instances and the DAS	101
	Default Synchronization for Files and Directories	101
	▼ To Resynchronize an Instance and the DAS Online	104

▼ To Resynchronize Library Files	106
▼ To Resynchronize Custom Configuration Files for an Instance	108
▼ To Resynchronize Users' Changes to Files	110
▼ To Resynchronize Additional Configuration Files	110
▼ To Prevent Deletion of Application-Generated Files	111
▼ To Resynchronize an Instance and the DAS Offline	111
Migrating EJB Timers	114
▼ To Enable Automatic EJB Timer Migration for Failed Clustered Instances	114
▼ To Migrate EJB Timers Manually	114
6 Administering Named Configurations	117
About Named Configurations	117
Types of Named Configurations	117
The default-config Configuration	118
Automatically Created Configurations	118
Directory for Configuration Synchronization	118
Creating, Listing, and Deleting Named Configurations	119
▼ To Create a Named Configuration	119
▼ To List the Named Configurations in a Domain	120
▼ To List the Targets of a Named Configuration	121
▼ To Delete a Named Configuration	122
Modifying Properties for Named Configurations and Instances	123
Properties for Port Numbers in a Named Configuration	124
▼ To Modify a Named Configuration's Properties	125
▼ To Modify Port Numbers of an Instance	126
7 Configuring HTTP Load Balancing	129
Setting Up HTTP Load Balancing	129
Prerequisites for Setting Up HTTP Load Balancing	129
Configuring GlassFish Server with Apache HTTP Server and mod_jk	130
HTTP Load Balancer Deployments	132
8 Upgrading Applications Without Loss of Availability	133
Application Compatibility	134

Upgrading In a Single Cluster	134
▼ To Upgrade an Application in a Single Cluster	134
Upgrading in Multiple Clusters	135
▼ To Upgrade a Compatible Application in Two or More Clusters	135
Upgrading Incompatible Applications	136
▼ To Upgrade an Incompatible Application by Creating a Second Cluster	137
 9 Configuring High Availability Session Persistence and Failover	139
Overview of Session Persistence and Failover	139
Requirements	140
Restrictions	140
Scope	141
Enabling the High Availability Session Persistence Service	142
▼ To Enable Availability for a Cluster, Standalone Instance or Container	142
Configuring Availability for Individual Web Applications	144
Configuring Replication and Multi-Threaded Concurrent Access to HttpSessionS	145
Using Single Sign-on with Session Failover	145
Using Coherence*Web for HTTP Session Persistence	147
Stateful Session Bean Failover	147
Configuring Availability for the EJB Container	148
Configuring Availability for an Individual Application or EJB Module	149
Configuring Availability for an Individual Bean	149
Specifying Methods to Be Checkpointed	149
 10 Configuring Java Message Service High Availability	151
Using Message Queue Broker Clusters With GlassFish Server	151
About Message Queue Broker Clusters	152
Configuring GlassFish Clusters to Use Message Queue Broker Clusters	153
▼ To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster With Master Broker	154
▼ To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster of Peer Brokers	155
▼ To Change the Master Broker in an Embedded or Local Broker Cluster	156
▼ To Migrate Between Types of Embedded or Local Conventional Broker Clusters	156
▼ To Configure a GlassFish Cluster to Use a Local Enhanced Broker Cluster	156

▼ To Configure a GlassFish Cluster to Use a Remote Broker Cluster	157
Connection Failover	158
Load-Balanced Delivery to MDBs	160
11 RMI-IIOP Load Balancing and Failover	161
Overview	161
General Requirements for Configuring Load Balancing	161
Load Balancing Models	162
InitialContext Load Balancing	162
InitialContext Summary	162
InitialContext Algorithm	163
Enabling RMI-IIOP Hardware Load Balancing and Failover	163
Per-Request Load Balancing (PRLB)	166
PRLB Summary	166
Enabling Per-Request Load Balancing	167
Index	171

Examples

EXAMPLE 2-1	Setting the Home Directory for the MKS Toolkit User	34
EXAMPLE 2-2	Determining if the sshd Daemon Is Running on an Oracle Solaris System	36
EXAMPLE 2-3	Determining if the sshd Daemon Is Running on a Linux System	37
EXAMPLE 2-4	Setting Up Public Key Authentication Without Encryption	41
EXAMPLE 2-5	Setting Up Encrypted Public Key Authentication	44
EXAMPLE 2-6	Creating an Alias for the SSH User's Password	46
EXAMPLE 2-7	Copying a GlassFish Server Installation to Multiple Hosts	47
EXAMPLE 2-8	Removing GlassFish Server Software From Multiple Hosts	49
EXAMPLE 3-1	Creating an SSH Node	53
EXAMPLE 3-2	Listing Basic Information About All SSH Nodes in a Domain	54
EXAMPLE 3-3	Listing Detailed Information About All SSH Nodes in a Domain	54
EXAMPLE 3-4	Testing if an SSH Node Is Reachable	55
EXAMPLE 3-5	Deleting an SSH Node	56
EXAMPLE 3-6	Creating a CONFIG Node	57
EXAMPLE 3-7	Listing Basic Information About All CONFIG Nodes in a Domain	58
EXAMPLE 3-8	Listing Detailed Information About All CONFIG Nodes in a Domain	58
EXAMPLE 3-9	Deleting a CONFIG Node	59
EXAMPLE 3-10	Updating an SSH Node	60
EXAMPLE 3-11	Updating a CONFIG Node	61
EXAMPLE 3-12	Changing a CONFIG Node to an SSH Node	63
EXAMPLE 4-1	Checking the Health of Instances in a Cluster	71
EXAMPLE 4-2	Validating That Multicast Transport Is Available for a Cluster	72
EXAMPLE 4-3	Creating a Cluster	77
EXAMPLE 4-4	Creating a Cluster and Setting GMS Options	77
EXAMPLE 4-5	Listing All Clusters in a Domain	78
EXAMPLE 4-6	Listing All Clusters That Are Associated With a Node	78
EXAMPLE 4-7	Deleting a Cluster	79
EXAMPLE 5-1	Creating a Clustered Instance Centrally	84

EXAMPLE 5-2	Listing Basic Information About All GlassFish Server Instances in a Domain ..	85
EXAMPLE 5-3	Listing Detailed Information About All GlassFish Server Instances in a Domain	86
EXAMPLE 5-4	Deleting an Instance Centrally	87
EXAMPLE 5-5	Starting a Cluster	88
EXAMPLE 5-6	Stopping a Cluster	89
EXAMPLE 5-7	Starting an Individual Instance Centrally	90
EXAMPLE 5-8	Stopping an Individual Instance Centrally	90
EXAMPLE 5-9	Restarting an Individual Instance Centrally	91
EXAMPLE 5-10	Creating a Clustered Instance Locally Without Specifying a Node	95
EXAMPLE 5-11	Creating a Clustered Instance Locally	96
EXAMPLE 5-12	Deleting an Instance Locally	98
EXAMPLE 5-13	Starting an Individual Instance Locally	99
EXAMPLE 5-14	Stopping an Individual Instance Locally	100
EXAMPLE 5-15	Restarting an Individual Instance Locally	100
EXAMPLE 5-16	Resynchronizing an Instance and the DAS Online	105
EXAMPLE 5-17	Updating the Option for Locating a Configuration File	109
EXAMPLE 5-18	config-files File	111
EXAMPLE 5-19	Resynchronizing an Instance and the DAS Offline	113
EXAMPLE 5-20	Migrating an EJB Timer	115
EXAMPLE 6-1	Creating a Named Configuration	120
EXAMPLE 6-2	Listing All Named Configurations in a Domain	121
EXAMPLE 6-3	Listing the Targets of a Named Configuration	121
EXAMPLE 6-4	Deleting a Named Configuration	123
EXAMPLE 6-5	Modifying a Property of a Named Configuration	126
EXAMPLE 6-6	Modifying a Port Number for an Instance	128
EXAMPLE 7-1	httpd.conf File for Load Balancing	131
EXAMPLE 7-2	workers.properties File for Load Balancing	131
EXAMPLE 9-1	Example of an EJB Deployment Descriptor With Availability Enabled	149
EXAMPLE 9-2	Example of EJB Deployment Descriptor Specifying Methods Checkpointing	150
EXAMPLE 11-1	Setting Load-Balancing Weights for RMI-IIOP Weighted Round-Robin Load Balancing	166

Preface

This book describes the high-availability features in GlassFish Server, including converged load balancing, HTTP load balancing, clusters, session persistence and failover.

This preface contains information about and conventions for the entire GlassFish Server Open Source Edition (GlassFish Server) documentation set.

GlassFish Server 3.1 is developed through the GlassFish project open-source community at <http://glassfish.java.net/>. The GlassFish project provides a structured process for developing the GlassFish Server platform that makes the new features of the Java EE platform available faster, while maintaining the most important feature of Java EE: compatibility. It enables Java developers to access the GlassFish Server source code and to contribute to the development of the GlassFish Server. The GlassFish project is designed to encourage communication between Oracle engineers and the community.

Oracle GlassFish Server Documentation Set

TABLE P-1 Books in the GlassFish Server Documentation Set

Book Title	Description
<i>Release Notes</i>	Provides late-breaking information about the software and the documentation and includes a comprehensive, table-based summary of the supported hardware, operating system, Java Development Kit (JDK), and database drivers.
<i>Quick Start Guide</i>	Explains how to get started with the GlassFish Server product.
<i>Installation Guide</i>	Explains how to install the software and its components.
<i>Upgrade Guide</i>	Explains how to upgrade to the latest version of GlassFish Server. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
<i>Deployment Planning Guide</i>	Explains how to build a production deployment of GlassFish Server that meets the requirements of your system and enterprise.

TABLE P-1 Books in the GlassFish Server Documentation Set *(Continued)*

Book Title	Description
<i>Administration Guide</i>	Explains how to configure, monitor, and manage GlassFish Server subsystems and components from the command line by using the <code>asadmin(1M)</code> utility. Instructions for performing these tasks from the Administration Console are provided in the Administration Console online help.
<i>Security Guide</i>	Provides instructions for configuring and administering GlassFish Server security.
<i>Application Deployment Guide</i>	Explains how to assemble and deploy applications to the GlassFish Server and provides information about deployment descriptors.
<i>Application Development Guide</i>	Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the GlassFish Server. These applications follow the open Java standards model for Java EE components and application programmer interfaces (APIs). This guide provides information about developer tools, security, and debugging.
<i>Add-On Component Development Guide</i>	Explains how to use published interfaces of GlassFish Server to develop add-on components for GlassFish Server. This document explains how to perform <i>only</i> those tasks that ensure that the add-on component is suitable for GlassFish Server.
<i>Embedded Server Guide</i>	Explains how to run applications in embedded GlassFish Server and to develop applications in which GlassFish Server is embedded.
<i>High Availability Administration Guide</i>	Explains how to configure GlassFish Server to provide higher availability and scalability through failover and load balancing.
<i>Performance Tuning Guide</i>	Explains how to optimize the performance of GlassFish Server.
<i>Troubleshooting Guide</i>	Describes common problems that you might encounter when using GlassFish Server and explains how to solve them.
<i>Error Message Reference</i>	Describes error messages that you might encounter when using GlassFish Server.
<i>Reference Manual</i>	Provides reference information in man page format for GlassFish Server administration commands, utility commands, and related concepts.
<i>Message Queue Release Notes</i>	Describes new features, compatibility issues, and existing bugs for Open Message Queue.
<i>Message Queue Technical Overview</i>	Provides an introduction to the technology, concepts, architecture, capabilities, and features of the Message Queue messaging service.
<i>Message Queue Administration Guide</i>	Explains how to set up and manage a Message Queue messaging system.

TABLE P-1 Books in the GlassFish Server Documentation Set (Continued)

Book Title	Description
<i>Message Queue Developer's Guide for JMX Clients</i>	Describes the application programming interface in Message Queue for programmatically configuring and monitoring Message Queue resources in conformance with the Java Management Extensions (JMX).
<i>Message Queue Developer's Guide for Java Clients</i>	Provides information about concepts and procedures for developing Java messaging applications (Java clients) that work with GlassFish Server.
<i>Message Queue Developer's Guide for C Clients</i>	Provides programming and reference information for developers working with Message Queue who want to use the C language binding to the Message Queue messaging service to send, receive, and process Message Queue messages.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P-2 Default Paths and File Names

Placeholder	Description	Default Value
<i>as-install</i>	Represents the base installation directory for GlassFish Server. In configuration files, <i>as-install</i> is represented as follows: <code>\${com.sun.aas.installRoot}</code>	Installations on the Oracle Solaris operating system, Linux operating system, and Mac OS operating system: <i>user's-home-directory/glassfish3/glassfish</i> Windows, all installations: <i>SystemDrive:\glassfish3\glassfish</i>
<i>as-install-parent</i>	Represents the parent of the base installation directory for GlassFish Server.	Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: <i>user's-home-directory/glassfish3</i> Windows, all installations: <i>SystemDrive:\glassfish3</i>
<i>domain-root-dir</i>	Represents the directory in which a domain is created by default.	<i>as-install/domains/</i>
<i>domain-dir</i>	Represents the directory in which a domain's configuration is stored. In configuration files, <i>domain-dir</i> is represented as follows: <code>\${com.sun.aas.instanceRoot}</code>	<i>domain-root-dir/domain-name</i>

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P-3 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name% su</code> Password:
<i>AaBbCc123</i>	A placeholder to be replaced with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the <i>User's Guide</i> . <i>A cache</i> is a copy that is stored locally. <i>Do not</i> save the file.

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P-4 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	<code>ls [-l]</code>	The <code>-l</code> option is not required.
{ }	Contains a set of choices for a required command option.	<code>-d {y n}</code>	The <code>-d</code> option requires that you use either the <code>y</code> argument or the <code>n</code> argument.
<code>\${ }</code>	Indicates a variable reference.	<code>\${com.sun.javaRoot}</code>	References the value of the <code>com.sun.javaRoot</code> variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.

TABLE P-4 Symbol Conventions (Continued)

Symbol	Description	Example	Meaning
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Documentation, Support, and Training

The Oracle web site provides information about the following additional resources:

- Documentation (<http://www.oracle.com/technetwork/indexes/documentation/index.html>)
- Support (<http://www.oracle.com/us/support/index.html>)
- Training (<http://education.oracle.com/>)

High Availability in GlassFish Server

This chapter describes the high availability features in GlassFish Server Open Source Edition 3.1.

The following topics are addressed here:

- [“Overview of High Availability” on page 17](#)
- [“How GlassFish Server Provides High Availability” on page 19](#)
- [“Recovering from Failures” on page 21](#)
- [“More Information” on page 24](#)

Overview of High Availability

High availability applications and services provide their functionality continuously, regardless of hardware and software failures. To make such reliability possible, GlassFish Server provides mechanisms for maintaining *application state data* between clustered GlassFish Server instances. Application state data, such as HTTP session data, stateful EJB sessions, and dynamic cache information, is replicated in real time across server instances. If any one server instance goes down, the session state is available to the next *failover* server, resulting in minimum application downtime and enhanced transactional security.

GlassFish Server provides the following high availability features:

- [“High Availability Session Persistence” on page 18](#)
- [“High Availability Java Message Service” on page 18](#)
- [“RMI-IIOP Load Balancing and Failover” on page 19](#)

Load Balancing With the Apache mod_jk Module

A common load balancing configuration for GlassFish Server 3.1 is to use the Apache HTTP 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 HTTP Server and mod_jk” on page 130](#) 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 19](#). 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:

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 151](#).

Connection Failover

The use of Message Queue broker clusters allows connection failover in the event of a broker failure. If the primary JMS host (Message Queue broker) in use by a GlassFish instance fails, connections to the failed JMS host will automatically fail over to another host in the JMS host list, allowing messaging operations to continue and maintaining JMS messaging semantics.

For more information about JMS connection failover, see [“Connection Failover” on page 158](#).

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.”](#)

How GlassFish Server Provides High Availability

GlassFish Server provides high availability through the following subcomponents and features:

- [“Storage for Session State Data” on page 20](#)
- [“Highly Available Clusters” on page 20](#)

Storage for Session State Data

Storing session state data enables the session state to be recovered after the failover of a server instance in a cluster. Recovering the session state enables the session to continue without loss of information. GlassFish Server supports in-memory session replication on other servers in the cluster for maintaining HTTP session and stateful session bean data.

In-memory session replication is implemented in GlassFish Server 3.1 as an OSGi module. Internally, the replication module uses a consistent hash algorithm to pick a replica server instance within a cluster of instances. This allows the replication module to easily locate the replica or replicated data when a container needs to retrieve the data.

The use of in-memory replication requires the Group Management Service (GMS) to be enabled. For more information about GMS, see [“Group Management Service” on page 66](#).

If server instances in a cluster are located on different hosts, ensure that the following prerequisites are met:

- To ensure that GMS and in-memory replication function correctly, the hosts must be on the same subnet.
- To ensure that in-memory replication functions correctly, the system clocks on all hosts in the cluster must be synchronized as closely as possible.

Highly Available Clusters

A *highly available cluster* integrates a state replication service with clusters and load balancer.

Note – When implementing a highly available cluster, use a load balancer that includes session-based stickiness as part of its load-balancing algorithm. Otherwise, session data can be misdirected or lost. An example of a load balancer that includes session-based stickiness is the Loadbalancer Plug-In available in Oracle GlassFish Server.

Clusters, Instances, Sessions, and Load Balancing

Clusters, server instances, load balancers, and sessions are related as follows:

- A server instance is not required to be part of a cluster. However, an instance that is not part of a cluster cannot take advantage of high availability through transfer of session state from one instance to other instances.
- The server instances within a cluster can be hosted on one or multiple hosts. You can group server instances across different hosts into a cluster.
- A particular load balancer can forward requests to server instances on multiple clusters. You can use this ability of the load balancer to perform an online upgrade without loss of service. For more information, see [“Upgrading in Multiple Clusters” on page 135](#).

- A single cluster can receive requests from multiple load balancers. If a cluster is served by more than one load balancer, you must configure the cluster in exactly the same way on each load balancer.
- Each session is tied to a particular cluster. Therefore, although you can deploy an application on multiple clusters, session failover will occur only within a single cluster.

The cluster thus acts as a safe boundary for session failover for the server instances within the cluster. You can use the load balancer and upgrade components within the GlassFish Server without loss of service.

SSH for Centralized Cluster Administration

GlassFish Server uses secure shell (SSH) to ensure that clusters that span multiple hosts can be administered centrally. To perform administrative operations on GlassFish Server instances that are remote from the domain administration server (DAS), the DAS must be able to communicate with those instances. If an instance is running, the DAS connects to the running instance directly. For example, when you deploy an application to an instance, the DAS connects to the instance and deploys the application to the instance.

However, the DAS cannot connect to an instance to perform operations on an instance that is not running, such as creating or starting the instance. For these operations, the DAS uses SSH to contact a remote host and administer instances there. SSH provides confidentiality and security for data that is exchanged between the DAS and remote hosts.

Note – The use of SSH to enable centralized administration of remote instances is optional. If the use of SSH is not feasible in your environment, you can administer remote instances locally.

For more information, see [Chapter 2, “Setting Up SSH for Centralized Administration.”](#)

Recovering from Failures

You can use various techniques to manually recover individual subcomponents after hardware failures such as disk crashes.

The following topics are addressed here:

- “Recovering the Domain Administration Server” on page 22
- “Recovering GlassFish Server Instances” on page 22
- “Recovering the HTTP Load Balancer and Web Server” on page 22
- “Recovering Message Queue” on page 22

Recovering the Domain Administration Server

Loss of the Domain Administration Server (DAS) affects only administration. GlassFish Server clusters and standalone instances, and the applications deployed to them, continue to run as before, even if the DAS is not reachable

Use any of the following methods to recover the DAS:

- Back up the domain periodically, so you have periodic snapshots. After a hardware failure, re-create the DAS on a new host, as described in “Re-Creating the Domain Administration Server (DAS)” in *GlassFish Server Open Source Edition 3.1 Administration Guide*.
- Put the domain installation and configuration on a shared and robust file system (NFS for example). If the primary DAS host fails, a second host is brought up with the same IP address and will take over with manual intervention or user supplied automation.
- Zip the GlassFish Server installation and domain root directory. Restore it on the new host, assigning it the same network identity.

Recovering GlassFish Server Instances

GlassFish Server provide tools for backing up and restoring GlassFish Server instances. For more information, see [“To Resynchronize an Instance and the DAS Offline” on page 111](#).

Recovering the HTTP Load Balancer and Web Server

There are no explicit commands to back up only a web server configuration. Simply zip the web server installation directory. After failure, unzip the saved backup on a new host with the same network identity. If the new host has a different IP address, update the DNS server or the routers.

Note – This assumes that the web server is either reinstalled or restored from an image first.

The Load Balancer Plug-In (plugins directory) and configurations are in the web server installation directory, typically `/opt/SUNWwbsvr`. The `web-install/web-instance/config` directory contains the `loadbalancer.xml` file.

Recovering Message Queue

When a Message Queue broker becomes unavailable, the method you use to restore the broker to operation depends on the nature of the failure that caused the broker to become unavailable:

- Power failure or failure other than disk storage

- Failure of disk storage

Additionally, the urgency of restoring an unavailable broker to operation depends on the type of the broker:

- **Standalone Broker.** When a standalone broker becomes unavailable, both service availability and data availability are interrupted. Restore the broker to operation as soon as possible to restore availability.
- **Broker in a Conventional Cluster.** When a broker in a conventional cluster becomes unavailable, service availability continues to be provided by the other brokers in the cluster. However, data availability of the persistent data stored by the unavailable broker is interrupted. Restore the broker to operation to restore availability of its persistent data.
- **Broker in an Enhanced Cluster.** When a broker in an enhanced cluster becomes unavailable, service availability and data availability continue to be provided by the other brokers in the cluster. Restore the broker to operation to return the cluster to its previous capacity.

Recovering From Power Failure and Failures Other Than Disk Storage

When a host is affected by a power failure or failure of a non-disk component such as memory, processor or network card, restore Message Queue brokers on the affected host by starting the brokers after the failure has been remedied.

To start brokers serving as Embedded or Local JMS hosts, start the GlassFish instances the brokers are servicing. To start brokers serving as Remote JMS hosts, use the `imqbrokerd` Message Queue utility.

Recovering from Failure of Disk Storage

Message Queue uses disk storage for software, configuration files and persistent data stores. In a default GlassFish installation, all three of these are generally stored on the same disk: the Message Queue software in *as-install-parent/mq*, and broker configuration files and persistent data stores (except for the persistent data stores of enhanced clusters, which are housed in highly available databases) in *as-install-parent/glassfish/domains/domain1/imq*. If this disk fails, restoring brokers to operation is impossible unless you have previously created a backup of these items. To create such a backup, use a utility such as `zip`, `gzip` or `tar` to create archives of these directories and all their content. When creating the backup, you should first quiesce all brokers and physical destinations, as described in “Quiescing a Broker” and “Pausing and Resuming a Physical Destination” in the *Message Queue 4.5 Administration Guide*, respectively. Then, after the failed disk is replaced and put into service, expand the backup archive into the same location.

Restoring the Persistent Data Store From Backup. For many messaging applications, restoring a persistent data store from backup does not produce the desired results because the backed up store does not represent the content of the store when the disk failure occurred. In

some applications, the persistent data changes rapidly enough to make backups obsolete as soon as they are created. To avoid issues in restoring a persistent data store, consider using a RAID or SAN data storage solution that is fault tolerant, especially for data stores in production environments.

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

Setting Up SSH for Centralized Administration

GlassFish Server uses secure shell (SSH) to ensure that clusters that span multiple hosts can be administered centrally. To perform administrative operations on GlassFish Server instances that are remote from the domain administration server (DAS), the DAS must be able to communicate with those instances. If an instance is running, the DAS connects to the running instance directly. For example, when you deploy an application to an instance, the DAS connects to the instance and deploys the application to the instance.

However, the DAS cannot connect to an instance to perform operations on an instance that is not running, such as creating or starting the instance. For these operations, the DAS uses SSH to contact a remote host and administer instances there. SSH provides confidentiality and security for data that is exchanged between the DAS and remote hosts.

Note – The use of SSH to enable centralized administration of remote instances is optional. If the use of SSH is not feasible in your environment, you can administer remote instances locally.

The following topics are addressed here:

- [“About SSH for Centralized Administration” on page 26](#)
- [“Setting Up Cygwin SSH on Windows” on page 28](#)
- [“Setting Up the MKS Toolkit on Windows” on page 31](#)
- [“Setting Up SSH on UNIX and Linux Systems” on page 35](#)
- [“Testing the SSH Setup on a Host” on page 38](#)
- [“Setting Up SSH User Authentication” on page 39](#)
- [“Installing and Removing GlassFish Server Software on Multiple Hosts” on page 46](#)

About SSH for Centralized Administration

In a typical GlassFish Server deployment, the DAS acts as the SSH client, and hosts where instances reside act as SSH servers. The SSH Server Daemon `sshd` must be running on hosts where instances reside, but is not required to be running on the DAS host. The DAS uses its own SSH client for communicating with hosts where instances reside. However, to generate keys and test SSH setup, a native SSH client must be installed on the DAS host.

Determining Whether to Use SSH for Centralized Administration

The use of SSH to enable centralized administration of remote instances is optional and is required only for specific operations. Instances local to the DAS can be administered without SSH. If SSH is not practicable in your environment, you can administer remote instances locally.

The requirements for SSH configuration and user management are different for each operating system on which GlassFish Server is supported. Therefore, the use of SSH for centralized administration involves using SSH tools to configure SSH on the operating system that you are using. Before setting up a GlassFish Server cluster, use the following considerations to determine whether to use SSH:

- If you are planning a large cluster of many instances, consider setting up SSH to enable centralized administration of the cluster. SSH simplifies the administration of the cluster by enabling you to perform all administrative operations on the cluster and its instances from the DAS. On UNIX and Linux systems, SSH is typically installed and preconfigured, and requires minimal additional setup. On Windows systems, additional setup is required to install and configure an SSH provider.
- If you are planning a small cluster of few instances, consider whether setting up SSH requires more effort than logging in to individual hosts to administer remote instances locally.

How you administer instances and the nodes on which they resides varies depending on whether SSH is available. The following table provides cross-references to instructions for administering nodes and instances depending on whether SSH is available.

SSH Availability	Node Administration Instructions	Instance Administration Instructions
Available	“Creating, Listing, Testing, and Deleting SSH Nodes” on page 52	“Administering GlassFish Server Instances Centrally” on page 82
Not available	“Creating, Listing, and Deleting CONFIG Nodes” on page 56	“Administering GlassFish Server Instances Locally” on page 92

Obtaining SSH Software

On UNIX and Linux systems, SSH software is typically installed as part of the base operating system.

However, on Windows systems, you must install one of the following SSH providers:

- Cygwin (<http://www.cygwin.com/>) release 1.7.6
- MKS Toolkit for Developers (<http://www.mkssoftware.com>) release 9.2

Determining the SSH User

Before setting up SSH, decide which SSH user GlassFish Server will use when connecting to remote hosts. For the following reasons, administration is simplest if the SSH user is the user that starts the DAS:

- For public key authentication, the user that starts the DAS must be able to read the SSH user's private key file.
- Remote instances are started as the SSH user.
- By default, the DAS assumes that the SSH user is the user that is running the DAS.

Requirements for the SSH User's Environment

The environment of the SSH user on any remote host to which the user will connect must meet the requirements that are stated in “Paths and Environment Settings for the JDK Software” in *GlassFish Server Open Source Edition 3.1-3.1.1 Release Notes*.

The SSH user's environment on a host is set by the environment set-up files that are run when the user uses SSH to run a command on the host. You must ensure that these files set up the SSH user's environment correctly.

The files that are run when the user uses SSH to run a command are different than the files that are run when the user logs in to a host. For example, in the bash shell, `.profile` and `.bashrc` are run when the user logs in, but only `.bashrc` is run when the user runs a command. Therefore, in the bash shell, you must ensure that `.bashrc` contains the required environment settings for the SSH user.

File Access Permissions on UAC-Enabled Windows Systems

Note – The [User Account Control \(UAC\)](#) feature is available only on some versions of the Windows operating system, for example, Windows 7, Windows Vista, and Windows 2008.

You might be using a UAC-enabled Windows system and choose to store files for GlassFish Server instances in a directory other than the SSH user's home directory. In this situation, the SSH user must have native (that is, nonvirtual) read and write access to the file system where the instances are to be stored. The OS-level administrator has such access by default. You can also configure the system to grant such access to other users. For more information, see the documentation for the Windows operating system.

Setting Up Cygwin SSH on Windows

Set up Cygwin SSH on the DAS host and on all hosts where instances in your cluster will reside.

The following topics are addressed here:

- “To Download and Install Cygwin ” on page 28
- “To Set the Path for Windows and for the Cygwin Shell” on page 29
- “To Set the Home Directory for the Cygwin SSH User” on page 30
- “To Configure and Start the Cygwin SSH Server Daemon `sshd`” on page 30

▼ To Download and Install Cygwin

For centralized GlassFish Server administration, a basic Cygwin installation that includes the SSH client and the SSH server daemon `sshd` is sufficient. The default installation options are sufficient to create such a basic installation.

- 1 Log in as a user with Administrator privileges.
- 2 Create the folder `C:\cygwin`.
- 3 From the [Cygwin home page \(http://www.cygwin.com/\)](http://www.cygwin.com/), download and save the `setup.exe` file to your desktop.
- 4 Run the `setup.exe` file.
- 5 Select the default for the following options:
 - Install from Internet
 - Install Root Directory: `C:\cygwin`
 - Install for All Users
- 6 Specify a folder for the local package directory that is not the Cygwin root folder, for example, `C:\cygwin\packages`.

7 Specify the connection method.

For example, if the host is connected to the Internet through a proxy server, specify the proxy server.

8 Select the mirror site from which to download the software.**9 Select the openssh package for installation.**

a. Under the Net category, search for openssh.

b. Locate the openssh package and click Skip.

The package is selected.

c. Click Next.

Any unsatisfied dependencies are listed.

10 Leave the Select Required Packages option selected and click Next

The packages are installed.

11 Click Finish.

See Also For detailed information about installing Cygwin, see “Internet Setup” in *Cygwin User's Guide* (<http://cygwin.com/cygwin-ug-net/setup-net.html#internet-setup>).

▼ To Set the Path for Windows and for the Cygwin Shell

To enable GlassFish Server tools to find commands for SSH, each user's path for Windows and for the Cygwin shell must contain the following directories:

- The Cygwin bin directory, for example C:\cygwin\bin
- The bin directory of the JDK software

1 Log in as a user with Administrator privileges.

Logging in as a user with Administrator privileges ensures that the change applies to all users.

2 In the System Information control panel, click Advanced→Environment Variables.**3 Add the following directories to the Path environment variable:**

- The Cygwin bin directory, for example C:\cygwin\bin
- The bin directory of the JDK software

▼ To Set the Home Directory for the Cygwin SSH User

The SSH Server Daemon `sshd` locates a user's home directory from the configuration in the user database, not from environment variables such as `HOME`. To ensure that all GlassFish Server commands can run without errors, each SSH user must be configured to have a home directory.

Each user on a Windows host where SSH is set up potentially has two home directories:

- **Windows home directory.** GlassFish Server commands, which are run in a Windows command window, use the Windows home directory.
- **SSH home directory.** SSH commands, which are run in a shell such as `bash` or `ksh`, use the SSH home directory.

If these home directories are different, GlassFish Server and SSH each locate a user's `.ssh` directory in different directories. To simplify the set up of SSH, configure each user's home directory for SSH and Windows to be the same directory. A disadvantage of this approach is that the SSH home directory has spaces in its path name. Spaces in path names are cumbersome in the UNIX environment.

- 1 **Log in as a user with Administrator privileges.**
- 2 **In the `c:\cygwin\etc\passwd` file, edit the home directory setting for the SSH user to specify the user's home directory for Windows.**

▼ To Configure and Start the Cygwin SSH Server Daemon `sshd`

Before You Begin Ensure that the following prerequisites are met:

- A user account is created for each user that will log in to the host through SSH.
- A password is set for each user account.

The SSH server daemon `sshd` disallows authentication of any user for whose account a password is not set.

- 1 **Double-click the Cygwin icon.**
A Cygwin terminal is started.
- 2 **If necessary, set the password for your user account.**
 - a. **Run the `passwd` command as follows:**

```
$ passwd user-name
```

user-name

The user name for your account.

b. Type a password.

The password for your Windows account is also set.

3 Configure SSH on the host.

a. Run the `ssh-host-config` command.

```
$ ssh-host-config
```

Tip – If you are using Windows XP, specify the `-y` option of `ssh-host-config` to answer yes to all prompts. If you run `ssh-host-config` with the `-y` option, omit [Step b](#).

b. Ensure that the following options are set to yes in the file `/etc/ssh_config`:

- `StrictModes`
- `PubkeyAuthentication`

The file `/etc/ssh_config` can also be accessed as `/cygdrive/c/cygwin/etc/sshd_config`.

4 Start the SSH server daemon `sshd`.

```
$ net start sshd
```

5 Confirm that the SSH server daemon `sshd` is running.

```
$ cygrunsrv --query sshd
Service           : sshd
Display name      : CYGWIN sshd
Current State     : Running
Controls Accepted : Stop
Command           : /usr/sbin/sshd -D
```

Next Steps After you have completed the setup of SSH on a host, test the setup on the host as explained in [“Testing the SSH Setup on a Host” on page 38](#).

Setting Up the MKS Toolkit on Windows

Set up the MKS Toolkit on the DAS host and on all hosts where instances in your cluster will reside.

The following topics are addressed here:

- [“To Install the MKS Toolkit” on page 32](#)
- [“To Set the Path for Windows and for the MKS Toolkit Shell” on page 32](#)

- “To Set the Home Directory for the MKS Toolkit SSH User” on page 32
- “To Configure and Start the MKS Toolkit SSH Server Daemon `sshd`” on page 34

▼ To Install the MKS Toolkit

For centralized GlassFish Server administration, the default installation of the MKS Toolkit is sufficient.

- Follow the instructions in the MKS Toolkit product documentation to install OpenSSH from the MKS Toolkit with default installation options.

See Also For detailed information about installing MKS Toolkit, see “Installing MKS Toolkit” in *MKS Toolkit v9.4 Release Notes* (http://www.mkssoftware.com/docs/rn/relnotes_tk94.asp#install).

▼ To Set the Path for Windows and for the MKS Toolkit Shell

To enable GlassFish Server tools to find commands for SSH, each user's path for Windows and for the MKS Toolkit shell must contain the following directories:

- The MKS Toolkit bin directory, for example `C:\Program Files\MKS Toolkit\mksnt`
- The bin directory of the JDK software

The MKS Toolkit installer automatically adds the MKS Toolkit bin directory to the path. However, you must add the bin directory of the JDK software to the path yourself.

- 1 **Log in as a user with Administrator privileges.**
Logging in as a user with Administrator privileges ensures that the change applies to all users.
- 2 **In the System Information control panel, click Advanced → Environment Variables.**
- 3 **Add the bin directory of the JDK software to the Path environment variable.**

▼ To Set the Home Directory for the MKS Toolkit SSH User

The SSH Server Daemon `sshd` locates a user's home directory from the configuration in the user database, not from environment variables such as `HOME`. To ensure that all GlassFish Server commands can run without errors, each SSH user must be configured to have a home directory.

Each user on a Windows host where SSH is set up potentially has two home directories:

- **Windows home directory.** GlassFish Server commands, which are run in a Windows command window, use the Windows home directory.
- **SSH home directory.** SSH commands, which are run in a shell such as bash or ksh, use the SSH home directory.

If these home directories are different, GlassFish Server and SSH each locate a user's `.ssh` directory in different directories. To simplify the set up of SSH, configure each user's home directory for SSH and Windows to be the same directory. A disadvantage of this approach is that the SSH home directory has spaces in its path name. Spaces in path names are cumbersome in the UNIX environment.

1 Compare the pairs of settings for Windows and the MKS Toolkit that are listed in the following table.

Windows Environment Variable	MKS Toolkit Field
HOMEPATH	Home Directory
HOMEDRIVE	Home Directory Drive

a. In a Windows command window, determine the values of the following environment variables:

- HOMEPATH
- HOMEDRIVE

b. In an MKS Toolkit shell, determine the current settings of the following fields for the user:

- Home Directory
- Home Directory Drive

```
$ userinfo user-name
```

user-name

The user name for the user whose home directory you are setting, for example Administrator.

2 If the settings do not match, update setting of each MKS Toolkit field to match its corresponding Windows environment variable.

If the settings match, no further action is required.

To update the settings, run the following command in an MKS Toolkit shell:

```
$ userinfo -u -fHomeDirDrive:"drive" -fHomeDir:"path" user-name
```

drive

The drive identifier of the disk drive on which the user's Windows home directory resides, for example, C:.

path

The path to the user's Windows home directory, for example, \Documents and Settings\Administrator.

user-name

The user name for the user whose home directory you are setting, for example Administrator.

Note – Do not set the HOME environment variable explicitly. If Home Directory and Home Directory Drive are set correctly, the HOME environment variable specifies the correct path by default.

3 In an MKS Toolkit shell, confirm that the settings were updated.

```
$ userinfo user-name
```

user-name

The user name for the user whose home directory you are setting, for example Administrator.

4 Log out of the host and log in to the host again.

5 Confirm that the home directories are the same as explained in [Step 1](#).

Example 2–1 Setting the Home Directory for the MKS Toolkit User

This example sets the home directory for the MKS Toolkit user Administrator to C:\Documents and Settings\Administrator.

```
$ userinfo -u -fHomeDirDrive:"C:"  
-fHomeDir:"\Documents and Settings\Administrator" Administrator
```

▼ To Configure and Start the MKS Toolkit SSH Server Daemon sshd

Note – Do *not* set the command shell to cmd.exe. The use of SSH for centralized GlassFish Server administration requires a shell in the style of a UNIX shell.

1 From the Programs menu, choose MKS Toolkit→Configuration→Configuration Information.

- 2 **Enable password authentication and strict modes.**
 - a. Click the Secure Shell Service tab.
 - b. Select the Password Authentication option.
 - c. Click Advanced settings.
 - d. Click the Login tab.
 - e. Deselect the Strict Modes option.
- 3 **If you are using SSH key-file authentication, enable MKSAUTH password authentication.**
 - a. Click the Authentication tab.
 - b. Under Enable/Disable Password using MKSAUTH, type the user's password and click the Enable.
- 4 **Start the SSH server daemon sshd.**
- 5 **Confirm that the SSH server daemon sshd is running.**

```
$ service query MKSSecureSH
Name:           MKS Secure Shell Service
Service Type:   WIN32_OWN_PROCESS
Current State:  RUNNING
Controls Accepted:  ACCEPT_STOP
Check Point:    0
Wait Hint:      0
Start Type:     AUTO_START
Error Control:  IGNORE
Path:           "C:\Program Files\MKS Toolkit\bin\secshd.exe"
Dependency:     NuTCRACKERSERVICE
Dependency:     tcpip
Service Start Name:  LocalSystem
```

Next Steps After you have completed the setup of SSH on a host, test the setup on the host as explained in [“Testing the SSH Setup on a Host” on page 38.](#)

Setting Up SSH on UNIX and Linux Systems

Setting up SSH on UNIX and Linux systems involves verifying that the SSH server daemon `sshd` is running and, if necessary, starting this daemon. Set up SSH on the DAS host and on all hosts where instances in your cluster will reside.

On UNIX and Linux systems, SSH software is typically installed as part of the base operating system. If SSH is not installed, download and install the appropriate [OpenSSH](http://www.openssh.com/) (<http://www.openssh.com/>) SSH package for your operating system.

How to set up SSH on UNIX and Linux systems depends on the flavor of the operating system that you are running, as explained in the following sections:

- “To Set Up SSH on Oracle Solaris Systems” on page 36
- “To Set Up SSH on MacOS Systems” on page 37
- “To Set Up SSH on Linux systems” on page 37

▼ To Set Up SSH on Oracle Solaris Systems

- 1 Ensure that the following options in the configuration file `/etc/ssh/sshd_config` are set to **yes**:

- `StrictModes`
- `PubkeyAuthentication`

- 2 Determine if the SSH server daemon `sshd` is running.

```
$ /usr/bin/svcs ssh
```

- 3 If the SSH server daemon `sshd` is not running, start this daemon.

If the daemon is running, no further action is required.

```
$ /usr/sbin/svcadm enable ssh
```

Example 2–2 Determining if the `sshd` Daemon Is Running on an Oracle Solaris System

This example confirms that the SSH server daemon `sshd` is running on an Oracle Solaris system.

```
$ /usr/bin/svcs ssh
STATE      STIME      FMRI
online     Jul_06     svc:/network/ssh:default
```

See Also `svcs(1)`

Next Steps After you have completed the setup of SSH on a host, test the setup on the host as explained in “Testing the SSH Setup on a Host” on page 38.

▼ To Set Up SSH on MacOS Systems

- 1 **Open System Preferences and click Sharing.**
The Sharing window opens.
- 2 **Ensure that Remote Login is selected in the Service list.**
- 3 **Ensure that either of the following is allowed access:**
 - All Users
 - The user that running the DAS or instance

Next Steps After you have completed the setup of SSH on a host, test the setup on the host as explained in [“Testing the SSH Setup on a Host” on page 38](#).

▼ To Set Up SSH on Linux systems

- 1 **Ensure that the following options in the configuration file `/etc/ssh/sshd_config` are set to yes:**
 - StrictModes
 - PubkeyAuthentication
- 2 **Determine if the SSH server daemon `sshd` is running.**
\$ `/sbin/service sshd status`
- 3 **If the SSH server daemon `sshd` is not running, start this daemon.**
If the daemon is running, no further action is required.
\$ `/sbin/service sshd start`

Example 2-3 Determining if the `sshd` Daemon Is Running on a Linux System

This example confirms that the SSH server daemon `sshd` is running on a Linux system.

```
$ /sbin/service sshd status
openssh-daemon (pid 2373) is running...
```

Next Steps After you have completed the setup of SSH on a host, test the setup on the host as explained in [“Testing the SSH Setup on a Host” on page 38](#).

Testing the SSH Setup on a Host

After setting up SSH on a host, test the setup to ensure that you can use SSH to contact the host from another host. Testing the SSH setup on a host verifies that the SSH server daemon `sshd` is running and that the SSH user has a valid user account on the host.

If you cannot use SSH to contact the host, troubleshoot the SSH setup before setting up SSH user authentication.

▼ To Test the SSH Setup on a Host

- 1 **From another host, use SSH to log in into the host that you are testing as the SSH user.**

```
$ ssh -l user-name host-name
```

user-name

The user name for the SSH user's account on the host.

host-name

The host name of the host that you are logging in to.

- 2 **In response to the prompt, type your password.**

If this step succeeds, your setup of SSH is complete.

The first time that you connect to a host, you might be warned that the authenticity cannot be established and be asked if you want to continue connection. If you trust the host, answer yes to connect to the host.

Troubleshooting

To obtain diagnostic information, use the `-v` option of the command-line SSH client and the `-d` option of the SSH server daemon `sshd`. How to start the SSH server daemon `sshd` manually depends on the operating system and SSH provider that you are using.

If the SSH server daemon `sshd` is set up on a host that has a firewall, ensure that a rule is defined to allow inbound traffic on the SSH port. The default SSH port is port 22.

If your connection is refused, the SSH server daemon `sshd` is not running and you must start the daemon. For instructions, see the following sections:

- [“To Configure and Start the Cygwin SSH Server Daemon `sshd`” on page 30](#)
- [“To Configure and Start the MKS Toolkit SSH Server Daemon `sshd`” on page 34](#)
- [“To Set Up SSH on Oracle Solaris Systems” on page 36](#)

If your connection is accepted, but you cannot log in, ensure that the SSH user has a valid user account on the host.

Next Steps After testing the SSH setup, set up SSH user authentication to enable SSH to authenticate users without prompting for a password. For more information, see [“Setting Up SSH User Authentication” on page 39](#).

Setting Up SSH User Authentication

When a GlassFish Server subcommand uses SSH to log in to a remote host, GlassFish Server must be able to authenticate the SSH user. Setting up SSH user authentication ensures that this requirement is met.

Before setting up SSH user authentication, determine the authentication scheme to use. If SSH is already deployed at your site, the authentication scheme to use might already be chosen for you.

The following table lists the authentication schemes that GlassFish Server supports. The table also lists the advantages and disadvantages of each authentication scheme.

Authentication Scheme	Advantages	Disadvantages
Public key without encryption	GlassFish Server provides tools to simplify set up.	SSH must be configured to locate users' key files in the correct location. File access permissions for key files and the directory that contains the key files must be set correctly.
Public key with passphrase-protected encryption	This scheme is more secure than public key authentication without encryption.	SSH must be configured to locate users' key files in the correct location. File access permissions for key files and the directory that contains the key files must be set correctly. For each SSH user, GlassFish Server password aliases are required for the encryption passphrase.
Password	No SSH configuration is required to locate key files or to ensure that file access permissions are correct.	For each SSH user, GlassFish Server password aliases are required for the SSH password.

The following topics are addressed here:

- [“To Set Up Public Key Authentication Without Encryption” on page 40](#)
- [“To Set Up Encrypted Public Key Authentication” on page 42](#)
- [“To Set Up Password Authentication” on page 45](#)

▼ To Set Up Public Key Authentication Without Encryption

Use the `setup-ssh` subcommand in local mode to set up public key authentication without encryption. This subcommand enables you to set up public key authentication on multiple hosts in a single operation.

The `setup-ssh` subcommand generates a key pair and distributes the public key file to specified hosts. The private key file and the public key file are protected only by the file system's file access permissions. If you require additional security, set up public key authentication with passphrase-protected encryption as explained in [“To Set Up Encrypted Public Key Authentication” on page 42](#).

Before You Begin Ensure that the following prerequisites are met:

- SSH is set up on each host where you are setting up public key authentication. For more information, see the following sections:
 - [“Setting Up Cygwin SSH on Windows” on page 28](#)
 - [“Setting Up the MKS Toolkit on Windows” on page 31](#)
 - [“Setting Up SSH on UNIX and Linux Systems” on page 35](#)
- Only the SSH user has write access to the following files and directories on each host where you are setting up public key authentication:
 - The SSH user's home directory
 - The `~/.ssh` directory
 - The `authorized_key` file

If other users can write to these files and directories, the secure service might not trust the `authorized_key` file and might disallow public key authentication.

1 Generate an SSH key pair and distribute the public key file to the hosts where you are setting up public key authentication.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for setting up an SSH key, see the `setup-ssh(1)` help page.

```
asadmin> setup-ssh [--sshuser sshuser] host-list
```

sshuser

The SSH user for which you are generating the SSH key pair. If you are running the subcommand as the SSH user, you may omit this option.

host-list

A space-separated list of the names of the hosts where the SSH public key is to be distributed.

After generating the SSH key pair, the subcommand uses SSH to log in to each host in *host-list* as the SSH user to distribute the public key. Each time a password is required to log in to a host, you are prompted for the SSH user's password.

2 In response to each prompt for a password, type the SSH user's password.

Example 2–4 Setting Up Public Key Authentication Without Encryption

This example generates and sets up an SSH key for the user `gfuser` on the hosts `sua01` and `sua02`. The command is run by the user `gfuser`.

```
asadmin> setup-ssh --generatekey=true sua01 sua02
Enter SSH password for gfuser@sua01>
Created directory /home/gfuser/.ssh
/usr/bin/ssh-keygen successfully generated the identification /home/gfuser/.ssh/id_rsa
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sua01
Successfully connected to gfuser@sua01 using keyfile /home/gfuser/.ssh/id_rsa
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sua02
Successfully connected to gfuser@sua02 using keyfile /home/gfuser/.ssh/id_rsa
Command setup-ssh executed successfully.
```

Next Steps After setting up public key authentication, test the setup by using `ssh` to log in as the SSH user to each host where the public key was distributed. For each host, log in first with the unqualified host name and then with the fully qualified name. If SSH does *not* prompt for password, public key authentication is set up correctly on the host.

If you are prompted for a password, verify that the public key file was copied correctly to the SSH user's `authorized_keys` file.

Troubleshooting Setup might fail because file access permissions in the SSH user's home directory are too permissive. In this situation, ensure that the file access permissions in the SSH user's home directory meet the requirements for performing this procedure.

If you have set the file access permissions in the SSH user's home directory correctly, setup might still fail if you are using the MKS Toolkit. In this situation, correct the problem in one of the following ways:

- On each remote host, copy the public key file to the SSH user's `~/ .ssh` directory and import the file. To import the file, select the Secure Service tab in the MKS configuration GUI and click Passwordless.
- Disable strict modes.

See Also

- “Setting Up Cygwin SSH on Windows” on page 28
- “Setting Up the MKS Toolkit on Windows” on page 31
- “Setting Up SSH on UNIX and Linux Systems” on page 35
- `setup-ssh(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help setup-ssh` at the command line.

▼ To Set Up Encrypted Public Key Authentication

Encrypted key file authentication uses an encrypted private key file that is protected with a passphrase. This passphrase must be provided to use the private key to unlock the public key. If you require encrypted public key authentication, you must use the SSH utility `ssh-keygen` to generate an SSH key pair with an encrypted private key. You can then use the `setup-ssh` subcommand to distribute the public key file to specified hosts.

To use the encrypted key file, GlassFish Server requires the passphrase with which the key file was encrypted. To provide this passphrase securely to GlassFish Server, create a GlassFish Server password alias to represent the passphrase and store this alias in a password file that is passed to the `asadmin(1M)` utility.

Note – Only the options that are required to complete this task are provided in each step. For information about all the options for the commands and subcommands in this task, see their help pages or man pages.

Before You Begin Ensure that the following prerequisites are met:

- SSH is set up on each host where you are setting up public key authentication. For more information, see the following sections:
 - [“Setting Up Cygwin SSH on Windows” on page 28](#)
 - [“Setting Up the MKS Toolkit on Windows” on page 31](#)
 - [“Setting Up SSH on UNIX and Linux Systems” on page 35](#)
- Only the SSH user has write access to the following files and directories on each host where you are setting up public key authentication:
 - The SSH user's home directory
 - The `~/ .ssh` directory
 - The `authorized_key` file

If other users can write to these files and directories, the secure service might not trust the `authorized_key` file and might disallow public key authentication.

1 Generate an SSH key pair with an encrypted private key file.

Use the SSH utility `ssh-keygen(1)` for this purpose.

```
$ ssh-keygen -t type
```

The algorithm that is to be used for the key and which must be `rsa`, `dsa`, or `rsa1`.

The `ssh-keygen` utility prompts you for a file in which to save the key.

2 To simplify the distribution of the key file, accept the default file.

The `ssh-keygen` utility prompts you for a passphrase.

3 In response to the prompt, type your choice of passphrase for encrypting the private key file.

The `ssh-keygen` utility prompts you to type the passphrase again.

4 In response to the prompt, type the passphrase that you set in [Step 3](#).**5 Distribute the public key file to the hosts where you are setting up public key authentication.**

Use the `setup-ssh(1) asadmin` subcommand for this purpose.

```
$ asadmin setup-ssh --generatekey=false host-list
host-list
```

A space-separated list of the names of the hosts where the SSH public key is to be distributed.

The subcommand uses SSH to log in to each host in *host-list* as the SSH user to distribute the public key. Each time a passphrase or a password is required to log in to a host, you are prompted for the passphrase or the SSH user's password.

6 In response to each prompt, type the requested information.

- In response to each prompt for a passphrase, type the passphrase that you set in [Step 3](#).
- In response to each prompt for a password, type the SSH user's password.

7 Create a GlassFish Server password alias for the passphrase that you set in [Step 3](#).

Use the `create-password-alias(1) asadmin` subcommand for this purpose.

```
$ asadmin create-password-alias alias-name
alias-name
```

Your choice of name for the alias that you are creating.

The `create-password-alias` subcommand prompts you to type the passphrase for which you are creating an alias.

8 In response to the prompt, type the passphrase that you set in [Step 3](#).

The `create-password-alias` subcommand prompts you to type the passphrase again.

9 In response to the prompt, type the passphrase that you set in [Step 3](#) again.**10 Create a plain text file that contains the following entry for the passphrase alias:**

```
AS_ADMIN_SSHKEYPASSPHRASE=${ALIAS=alias-name}
alias-name
```

The alias name that you specified in [Step 7](#).

Note – When you create an SSH node, pass this file as the `--passwordfile` option of the `asadmin` utility. For more information, see [“To Create an SSH Node” on page 52](#).

Example 2–5 Setting Up Encrypted Public Key Authentication

This example generates an SSH key pair with an encrypted private key for the user `gfadmin` and distributes the public key to the hosts `sj01` and `ja02`. The example also creates an alias that is named `ssh-key-passphrase` for the private key's passphrase.

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/gfadmin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gfadmin/.ssh/id_rsa.
Your public key has been saved in /home/gfadmin/.ssh/id_rsa.pub.
The key fingerprint is:
db:b5:f6:0d:fe:16:33:91:20:64:90:1a:84:66:f5:d0 gfadmin@dashost
$ asadmin setup-ssh --generatekey=false sj01 sj02
Key /home/gfadmin/.ssh/id_rsa is encrypted
Enter key passphrase>
Enter SSH password for gfadmin@sj01>
Copied keyfile /home/gfadmin/.ssh/id_rsa.pub to gfadmin@sj01
Successfully connected to gfadmin@sj01 using keyfile /home/gfadmin/.ssh/id_rsa
Successfully connected to gfadmin@sj02 using keyfile /home/gfadmin/.ssh/id_rsa
SSH public key authentication is already configured for gfadmin@sj02
Command setup-ssh executed successfully.
$ asadmin create-password-alias ssh-key-passphrase
Enter the alias password>
Enter the alias password again>
Command create-password-alias executed successfully.
```

The entry in the password file for the `ssh-key-passphrase` alias is as follows:

```
AS_ADMIN_SSHKEYPASSPHRASE=${ALIAS=ssh-key-passphrase}
```

Troubleshooting Setup might fail because file access permissions in the SSH user's home directory are too permissive. In this situation, ensure that the file access permissions in the SSH user's home directory meet the requirements for performing this procedure.

If you have set the file access permissions in the SSH user's home directory correctly, setup might still fail if you are using the MKS Toolkit. In this situation, correct the problem in one of the following ways:

- On each remote host, copy the public key file to the SSH user's `~/.ssh` directory and import the file. To import the file, select the Secure Service tab in the MKS configuration GUI and click Passwordless.
- Disable strict modes.

- See Also**
- “Setting Up Cygwin SSH on Windows” on page 28
 - “Setting Up the MKS Toolkit on Windows” on page 31
 - “Setting Up SSH on UNIX and Linux Systems” on page 35
 - `asadmin(1M)`
 - `create-password-alias(1)`
 - `setup-ssh(1)`
 - `ssh-keygen(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help create-password-alias`
- `asadmin help setup-ssh`

▼ To Set Up Password Authentication

To use SSH to log in to a remote host, GlassFish Server requires the SSH user's password. To provide this password securely to GlassFish Server, create a GlassFish Server password alias to represent the password and store this alias in a password file that is passed to the `asadmin(1M)` utility.

Before You Begin Ensure that SSH is set up on each host where you are setting up password authentication. For more information, see the following sections:

- “Setting Up Cygwin SSH on Windows” on page 28
- “Setting Up the MKS Toolkit on Windows” on page 31
- “Setting Up SSH on UNIX and Linux Systems” on page 35

1 Create an alias for the SSH user's password.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for creating a password alias, see the `create-password-alias(1)` help page.

```
asadmin> create-password-alias alias-name
alias-name
```

Your choice of name for the alias that you are creating.

The `create-password-alias` subcommand prompts you to type the password for which you are creating an alias.

2 In response to the prompt, type the SSH user's password.

The `create-password-alias` subcommand prompts you to type the password again.

- 3 In response to the prompt, type the SSH user's password again.
- 4 Create a plain text file that contains the following entry for the password alias:

```
AS_ADMIN_SSHPASSWORD=${ALIAS=alias-name}
```

alias-name

The alias name that you specified in [Step 1](#).

Note – When you create an SSH node, pass this file as the `--passwordfile` option of the `asadmin` utility. For more information, see [“To Create an SSH Node” on page 52](#).

Example 2–6 Creating an Alias for the SSH User's Password

This example creates an alias that is named `ssh-password` for the SSH user's password.

```
$ asadmin create-password-alias ssh-password
Enter the alias password>
Enter the alias password again>
Command create-password-alias executed successfully.
```

The entry in the password file for the `ssh-password` alias is as follows:

```
AS_ADMIN_SSHPASSWORD=${ALIAS=ssh-password}
```

- See Also**
- [“Setting Up Cygwin SSH on Windows” on page 28](#)
 - [“Setting Up the MKS Toolkit on Windows” on page 31](#)
 - [“Setting Up SSH on UNIX and Linux Systems” on page 35](#)
 - `asadmin(1M)`
 - `create-password-alias(1)`

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

Installing and Removing GlassFish Server Software on Multiple Hosts

GlassFish Server software must be installed on all hosts where GlassFish Server will run. How to install GlassFish Server software on multiple hosts depends on the degree of control that you require over the installation on each host.

- If you require complete control over the installation on each host, install the software from a GlassFish Server distribution on each host individually. For more information, see *GlassFish Server Open Source Edition 3.1 Installation Guide*.

- If the same set up on each host is acceptable, copy an existing GlassFish Server installation to the hosts. For more information, see [“To Copy a GlassFish Server Installation to Multiple Hosts” on page 47](#).

GlassFish Server also enables you to remove GlassFish Server software from multiple hosts in a single operation. For more information, see [“To Remove GlassFish Server Software From Multiple Hosts” on page 48](#).

The following topics are addressed here:

- [“To Copy a GlassFish Server Installation to Multiple Hosts” on page 47](#)
- [“To Remove GlassFish Server Software From Multiple Hosts” on page 48](#)

▼ To Copy a GlassFish Server Installation to Multiple Hosts

Use the `install-node` subcommand in local mode to copy an installation of GlassFish Server software to multiple hosts.

Before You Begin Ensure that SSH is set up on the host where you are running the subcommand and on each host where you are copying the GlassFish Server software.

● Run the `install-node` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for copying an installation of GlassFish Server software, see the `install-node(1)` help page.

```
asadmin> install-node host-list
```

host-list

A space-separated list of the names of the hosts where you are copying the installation of GlassFish Server software.

Example 2–7 Copying a GlassFish Server Installation to Multiple Hosts

This example copies the GlassFish Server software on the host where the subcommand is run to the default location on the hosts `sj03.example.com` and `sj04.example.com`.

```
asadmin> install-node sj03.example.com sj04.example.com
Created installation zip /home/gfuser/glassfish2339538623689073993.zip
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser
/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to
```

```
sj03.example.com:/export/glassfish3
Installing glassfish2339538623689073993.zip into sj03.example.com:/export/glassfish3
Removing sj03.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj03.example.com:/export/glassfish3/bin
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser
/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to
sj04.example.com:/export/glassfish3
Installing glassfish2339538623689073993.zip into sj04.example.com:/export/glassfish3
Removing sj04.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj04.example.com:/export/glassfish3/bin
Command install-node executed successfully
```

See Also `install-node(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help install-node` at the command line.

▼ To Remove GlassFish Server Software From Multiple Hosts

Use the `uninstall-node` subcommand in local mode to remove GlassFish Server software from multiple hosts.

Before You Begin Ensure that the following prerequisites are met:

- SSH is set up on the host where you are running the subcommand and on each host from which you are removing the GlassFish Server software.
- No process is accessing the parent of the base installation directory for the GlassFish Server software or any subdirectory of this directory.
- The configuration of the following items is the same on each host from which you are removing the GlassFish Server software:
 - Parent of the base installation directory for the GlassFish Server software
 - SSH port
 - SSH user
 - SSH key file

● Run the `uninstall-node` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for removing GlassFish Server software, see the `uninstall-node(1)` help page.

```
asadmin> uninstall-node host-list
```


host-list

A space-separated list of the names of the hosts from which you are removing GlassFish Server software.

Example 2-8 Removing GlassFish Server Software From Multiple Hosts

This example removes GlassFish Server software on the hosts `sj03.example.com` and `sj04.example.com` from the default location.

```
asadmin> uninstall-node sj03 sj04
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser
/.ssh/id_rsa
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser
/.ssh/id_rsa
Command uninstall-node executed successfully.
```

See Also `uninstall-node(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help uninstall-node` at the command line.

Administering GlassFish Server Nodes

A *node* represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside. A node's configuration contains information about the host such as the name of the host and the location where the GlassFish Server is installed on the host.

The following topics are addressed here:

- “Types of GlassFish Server Nodes” on page 51
- “Creating, Listing, Testing, and Deleting SSH Nodes” on page 52
- “Creating, Listing, and Deleting CONFIG Nodes” on page 56
- “Updating and Changing the Type of a Node” on page 60

Types of GlassFish Server Nodes

Each GlassFish Server node is one of the following types of node:

SSH

An SSH node supports communication over secure shell (SSH). If SSH is set up and you plan to administer your GlassFish Server instances centrally, the instances *must* reside on SSH nodes.

An SSH node's configuration contains the information that is required to connect to the host through SSH. This information includes, for example, the user name of the SSH user and the port number for SSH connections to the host.

CONFIG

A CONFIG node does not support remote communication. If SSH is not set up and you plan to administer your instances locally, the instances can reside on CONFIG nodes. You *cannot* use CONFIG nodes for instances that you plan to administer centrally.

Each domain contains a predefined CONFIG node that is named `localhost-domain`, where *domain* is the name of the domain. On the host where the domain administration server

(DAS) is running, this node represents the local host.

Creating, Listing, Testing, and Deleting SSH Nodes

An SSH node supports communication over SSH. If SSH is set up and you plan to administer your GlassFish Server instances centrally, the instances *must* reside on SSH nodes. For information about setting up SSH, see [Chapter 2, “Setting Up SSH for Centralized Administration.”](#)

GlassFish Server enables you to create SSH nodes for use by instances, obtain information about SSH nodes, test if SSH nodes are reachable, and delete SSH nodes that are no longer required.

The following topics are addressed here:

- “To Create an SSH Node” on page 52
- “To List SSH Nodes in a Domain” on page 53
- “To Test if an SSH Node is Reachable” on page 54
- “To Delete an SSH Node” on page 55

▼ To Create an SSH Node

Use the `create-node-ssh` subcommand in remote mode to create an SSH node.

Before You Begin Ensure that the user of the DAS can use SSH to log in to the host that the node will represent. By default, the `create-node-ssh` subcommand validates the node's parameters and the SSH connection to the host. If the user of the DAS cannot use SSH to log in to the host, the validation fails.

- 1 Ensure that the DAS is running.**
Remote subcommands require a running server.
- 2 Run the `create-node-ssh` subcommand.**

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the node, see the `create-node-ssh(1)` help page.

If you are using password authentication for the SSH user, you must specify a password file through the `--passwordfile` option of the `asadmin(1M)` utility. For more information about SSH user authentication, see [“Setting Up SSH User Authentication” on page 39.](#)

```
asadmin> create-node-ssh --nodehost node-host [--installdir install-dir ]
node-name
```

node-host

The name of the host that the node represents. The name of the host must be specified. Otherwise, an error occurs.

install-dir

The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, `/export/glassfish3/`. If the GlassFish Server software is installed in the same directory on the node's host and the DAS host, you can omit this option.

node-name

Your choice of name for the node that you are creating.

Example 3–1 Creating an SSH Node

This example creates the SSH node `sj01` to represent the host `sj01.example.com`. The GlassFish Server software is installed in the same directory on the DAS host and on the host `sj01.example.com`.

```
asadmin> create-node-ssh --nodehost sj01.example.com sj01
Command create-node-ssh executed successfully.
```

Troubleshooting The `create-node-ssh` subcommand might fail to create the node and report the error `Illegal sftp packet len`. If this error occurs, ensure that no the startup file on the remote host displays text for noninteractive shells. Examples of startup files are `.bashrc`, `.cshrc`, `.login`, and `.profile`.

The SSH session interprets any text message that is displayed during login as a file-transfer protocol packet. Therefore, any statement in a startup file that displays text messages corrupts the SSH session, causing this error.

See Also `create-node-ssh(1)`

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

Next Steps After creating a node, you can create instances on the node as explained in the following sections:

- [“To Create an Instance Centrally” on page 82](#)
- [“To Create an Instance Locally” on page 92](#)

▼ To List SSH Nodes in a Domain

Use the `list-nodes-ssh` subcommand in remote mode to obtain information about existing SSH nodes in a domain.

Note – To obtain information about all existing nodes in a domain, use the `list-nodes(1)` subcommand.

- 1 **Ensure that the DAS is running.**
Remote subcommands require a running server.
- 2 **Run the `list-nodes-ssh(1)` subcommand.**
`asadmin> list-nodes-ssh`

Example 3–2 Listing Basic Information About All SSH Nodes in a Domain

This example lists the name, type, and host of all SSH nodes in the current domain.

```
asadmin> list-nodes-ssh
sj01  SSH  sj01.example.com
sj02  SSH  sj02.example.com
Command list-nodes-ssh executed successfully.
```

Example 3–3 Listing Detailed Information About All SSH Nodes in a Domain

This example lists detailed information about all SSH nodes in the current domain.

```
asadmin> list-nodes-ssh --long=true
NODE NAME  TYPE  NODE HOST  INSTALL DIRECTORY  REFERENCED BY
sj01       SSH   sj01.example.com  /export/glassfish3  pmd-i1
sj02       SSH   sj02.example.com  /export/glassfish3  pmd-i2
Command list-nodes-ssh executed successfully.
```

- See Also**
- `list-nodes(1)`
 - `list-nodes-ssh(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help list-nodes`
- `asadmin help list-nodes-ssh`

▼ **To Test if an SSH Node is Reachable**

Use the `ping-node-ssh` subcommand in remote mode to test if an SSH node is reachable.

- Before You Begin** Ensure that SSH is configured on the host where the DAS is running and on the host that the node represents.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the ping-node-ssh subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for testing the node, see the `ping-node-ssh(1)` help page.

```
asadmin> ping-node-ssh node-name
```

node-name

The name of the node to test.

Example 3–4 Testing if an SSH Node Is Reachable

This example tests if the SSH node `sj01` is reachable.

```
asadmin> ping-node-ssh sj01
Successfully made SSH connection to node sj01 (sj01.example.com)
Command ping-node-ssh executed successfully.
```

See Also `ping-node-ssh(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help ping-node-ssh` at the command line.

▼ To Delete an SSH Node

Use the `delete-node-ssh` subcommand in remote mode to delete an SSH node.

Deleting a node removes the node from the configuration of the DAS. The node's directories and files are deleted when the last GlassFish Server instance that resides on the node is deleted.

Before You Begin Ensure that no GlassFish Server instances reside on the node that you are deleting. For information about how to delete an instance, see the following sections.

- [“To Delete an Instance Centrally” on page 86](#)
- [“To Delete an Instance Locally” on page 96](#)

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Confirm that no instances reside on the node that you are deleting.

```
asadmin> list-nodes-ssh --long=true
```

3 Run the `delete-node-ssh(1)` subcommand.

```
asadmin> delete-node-ssh node-name
```

node-name

The name of the node that you are deleting.

Example 3–5 Deleting an SSH Node

This example confirms that no instances reside on the SSH node `sj01` and deletes the node `sj01`.

```
asadmin> list-nodes-ssh --long=true
NODE NAME   TYPE   NODE HOST           INSTALL DIRECTORY   REFERENCED BY
sj01        SSH    sj01.example.com     /export/glassfish3
sj02        SSH    sj02.example.com     /export/glassfish3   pmd-i2
Command list-nodes-ssh executed successfully.
asadmin> delete-node-ssh sj01
Command delete-node-ssh executed successfully.
```

- See Also**
- [“To Delete an Instance Centrally” on page 86](#)
 - [“To Delete an Instance Locally” on page 96](#)
 - `delete-node-ssh(1)`
 - `list-nodes-ssh(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-node-ssh`
- `asadmin help list-nodes-ssh`

Creating, Listing, and Deleting CONFIG Nodes

A CONFIG node does not support remote communication. If SSH is not set up and you plan to administer your instances locally, the instances can reside on CONFIG nodes. You *cannot* use CONFIG nodes for instances that you plan to administer centrally.

GlassFish Server enables you to create CONFIG nodes for use by instances, obtain information about CONFIG nodes, and delete CONFIG nodes that are no longer required.

The following topics are addressed here:

- [“To Create a CONFIG Node” on page 56](#)
- [“To List CONFIG Nodes in a Domain” on page 58](#)
- [“To Delete a CONFIG Node” on page 59](#)

▼ To Create a CONFIG Node

Use the `create-node-config` command in remote mode to create a CONFIG node.

Note – If you create an instance locally on a host for which no nodes are defined, you can create the instance without creating a node beforehand. In this situation, GlassFish Server creates a CONFIG node for you. The name of the node is the unqualified name of the host. For more information, see [“To Create an Instance Locally” on page 92](#).

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `create-node-config` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the node, see the `create-node-config(1)` help page.

```
asadmin> create-node-config [--nodehost node-host] [--installdir install-dir ]
node-name
```

node-host

The name of the host that the node represents. You may omit this option. The name of the host can be determined when instances that reside on the node are created.

install-dir

The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, `/export/glassfish3/`. You may omit this option. The installation directory can be determined when instances that reside on the node are created.

node-name

Your choice of name for the node that you are creating.

Example 3–6 Creating a CONFIG Node

This example creates the CONFIG node `cfg01`. The host that the node represents and the installation directory of the GlassFish Server software on the host are to be determined when instances are added to the node.

```
asadmin> create-node-config cfg01
Command create-node-config executed successfully.
```

See Also `create-node-config(1)`

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

Next Steps After creating a node, you can create instances on the node as explained in [“To Create an Instance Locally” on page 92](#).

▼ To List CONFIG Nodes in a Domain

Use the `list-nodes-config` subcommand in remote mode to obtain information about existing CONFIG nodes in a domain.

Note – To obtain information about all existing nodes in a domain, use the `list-nodes(1)` subcommand.

- 1 Ensure that the DAS is running.**
Remote subcommands require a running server.
- 2 Run the `list-nodes-config(1)` subcommand.**
`asadmin> list-nodes-config`

Example 3-7 Listing Basic Information About All CONFIG Nodes in a Domain

This example lists the name, type, and host of all CONFIG nodes in the current domain.

```
asadmin> list-nodes-config
localhost-domain1 CONFIG localhost
cfg01 CONFIG cfg01.example.com
cfg02 CONFIG cfg02.example.com
Command list-nodes-config executed successfully.
```

Example 3-8 Listing Detailed Information About All CONFIG Nodes in a Domain

This example lists detailed information about all CONFIG nodes in the current domain.

```
asadmin> list-nodes-config --long=true
NODE NAME      TYPE      NODE HOST      INSTALL DIRECTORY  REFERENCED BY
localhost-domain1 CONFIG    localhost      /export/glassfish3
cfg01          CONFIG    cfg01.example.com /export/glassfish3  yml-i1
cfg02          CONFIG    cfg02.example.com /export/glassfish3  yml-i2
Command list-nodes-config executed successfully.
```

- See Also**
- `list-nodes(1)`
 - `list-nodes-config(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help list-nodes`

- `asadmin help list-nodes-config`

▼ To Delete a CONFIG Node

Use the `delete-node-config` subcommand in remote mode to delete a CONFIG node.

Deleting a node removes the node from the configuration of the DAS. The node's directories and files are deleted when the last GlassFish Server instance that resides on the node is deleted.

Before You Begin Ensure that no GlassFish Server instances reside on the node that you are deleting. For information about how to delete an instance that resides on a CONFIG node, see [“To Delete an Instance Locally” on page 96](#).

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Confirm that no instances reside on the node that you are deleting.

```
asadmin> list-nodes-config --long=true
```

3 Run the `delete-node-config(1)` subcommand.

```
asadmin> delete-node-config node-name
```

node-name

The name of the node that you are deleting.

Example 3–9 Deleting a CONFIG Node

This example confirms that no instances reside on the CONFIG node `cfg01` and deletes the node `cfg01`.

```
asadmin> list-nodes-config --long=true
NODE NAME          TYPE      NODE HOST          INSTALL DIRECTORY  REFERENCED BY
localhost-domain1  CONFIG    localhost          /export/glassfish3
cfg01              CONFIG    cfg01.example.com  /export/glassfish3
cfg02              CONFIG    cfg02.example.com  /export/glassfish3  yml-i2
Command list-nodes-config executed successfully.
asadmin> delete-node-config cfg01
Command delete-node-config executed successfully.
```

- See Also**
- [“To Delete an Instance Locally” on page 96](#)
 - `delete-node-config(1)`
 - `list-nodes-config(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-node-config`
- `asadmin help list-nodes-config`

Updating and Changing the Type of a Node

GlassFish Server enables you to update the configuration data of any node and to change the type of a node.

The following topics are addressed here:

- [“To Update an SSH Node” on page 60](#)
- [“To Update a CONFIG Node” on page 61](#)
- [“To Change the Type of a Node” on page 62](#)

▼ To Update an SSH Node

Use the `update-node-ssh` subcommand in remote mode to update an SSH node.

Options of this subcommand specify the new values of the node's configuration data. If you omit an option, the existing value is unchanged.

Before You Begin Ensure that the following prerequisites are met:

- SSH is configured on the host where the DAS is running and on the host that the node represents.
- The node that you are updating exists.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `update-node-ssh` subcommand.

```
asadmin> update-node-ssh options node-name
```

options

Options of the `update-node-ssh` subcommand for changing the node's configuration data.
For information about these options, see the `update-node-ssh(1)` help page.

node-name

The name of the SSH node to update.

Example 3–10 Updating an SSH Node

This example updates the host that the node `sj01` represents to `adc01.example.com`.

```
asadmin> update-node-ssh --nodehost adc01.example.com sj01
Command update-node-ssh executed successfully.
```

See Also `update-node-ssh(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help update-node-ssh` at the command line.

▼ To Update a CONFIG Node

Use the `update-node-config` subcommand in remote mode to update a CONFIG node.

Options of this subcommand specify the new values of the node's configuration data. If you omit an option, the existing value is unchanged.

Before You Begin Ensure that the node that you are updating exists.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `update-node-config` subcommand.

```
asadmin> uupdate-node-config options node-name
```

options

Options of the `update-node-config` subcommand for changing the node's configuration data. For information about these options, see the `update-node-config(1)` help page.

node-name

The name of the CONFIG node to update.

Example 3–11 Updating a CONFIG Node

This example updates the host that the node `cfg02` represents to `adc02.example.com`.

```
asadmin> update-node-config --nodehost adc02.example.com cfg02
Command update-node-config executed successfully.
```

See Also `update-node-config(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help update-node-config` at the command line.

▼ To Change the Type of a Node

The subcommands for updating a node can also be used to change a CONFIG node to an SSH node or to change an SSH node to a CONFIG node

Changing a CONFIG node to an SSH node enables SSH communication for the node. As part of this process, you can also change other configuration data for the node.

Options of the subcommands for updating a node specify the new values of the node's configuration data. For most options, if you omit the option, the existing value is unchanged. However, default values are applied if any of the following options of the `update-node-ssh` subcommand is omitted:

- `--sshport`
- `--sshuser`
- `--sshkeyfile`



Caution – Changing an SSH node to a CONFIG node disables remote communication for the node.

Before You Begin Ensure that the following prerequisites are met:

- SSH is configured on the host where the DAS is running and on the host that the node represents.
- The node the type of which you are changing exists.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `update-node-ssh` subcommand or the `update-node-config` subcommand, depending on the current type of the node.

- **To change a CONFIG node to an SSH node, run the `update-node-ssh` subcommand on the node.**

```
asadmin> update-node-ssh [options] config-node-name
```

options

Options of the `update-node-ssh` subcommand for changing the node's configuration data. For information about these options, see the `update-node-ssh(1)` help page.

config-node-name

The name of the CONFIG node to change.

- To change an SSH node to a CONFIG node, run the **update-node-config** subcommand on the node.

```
asadmin> update-node-config [options] ssh-node-name
```

options

Options of the update-node-config subcommand for changing the node's configuration data. For information about these options, see the update-node-config(1) help page.

ssh-node-name

The name of the SSH node to change.

Example 3–12 Changing a CONFIG Node to an SSH Node

This example changes the CONFIG node `cfg02` to an SSH node .

```
asadmin> update-node-ssh cfg02  
Command update-node-ssh executed successfully.
```

- See Also**
- update-node-config(1)
 - update-node-ssh(1)

You can also view the full syntax and options of the subcommand by typing the following commands at the command line.

- `asadmin help update-node-config`
- `asadmin help update-node-ssh`

Administering GlassFish Server Clusters

A *cluster* is a collection of GlassFish Server instances that work together as one logical entity. A cluster provides a runtime environment for one or more Java Platform, Enterprise Edition (Java EE) applications. A cluster provides high availability through failure protection, scalability, and load balancing.

The Group Management Service (GMS) enables instances to participate in a cluster by detecting changes in cluster membership and notifying instances of the changes. To ensure that GMS can detect changes in cluster membership, a cluster's GMS settings must be configured correctly.

The following topics are addressed here:

- [“About GlassFish Server Clusters” on page 65](#)
- [“Group Management Service” on page 66](#)
- [“Creating, Listing, and Deleting Clusters” on page 75](#)

About GlassFish Server Clusters

A *cluster* is a named collection of GlassFish Server instances that share the same applications, resources, and configuration information. For information about GlassFish Server instances, see [Chapter 5, “Administering GlassFish Server Instances.”](#)

GlassFish Server enables you to administer all the instances in a cluster as a single unit from a single host, regardless of whether the instances reside on the same host or different hosts. You can perform the same operations on a cluster that you can perform on an unclustered instance, for example, deploying applications and creating resources.

A cluster provides high availability through failure protection, scalability, and load balancing.

- **Failure protection.** If an instance or a host in a cluster fails, GlassFish Server detects the failure and recovers the user session state. If a load balancer is configured for the cluster, the load balancer redirects requests from the failed instance to other instances in the cluster.

Because the same applications and resources are on all instances in the cluster, an instance can fail over to any other instance in the cluster.

To enable the user session state to be recovered, each instance in a cluster sends in-memory state data to another instance. As state data is updated in any instance, the data is replicated.

- **Scalability.** If increased capacity is required, you can add instances to a cluster with no disruption in service. When an instance is added or removed, the changes are handled automatically.
- **Load balancing.** If instances in a cluster are distributed among different hosts, the workload can be distributed among the hosts to increase overall system throughput.

Group Management Service

The Group Management Service (GMS) is an infrastructure component that is enabled for the instances in a cluster. When GMS is enabled, if a clustered instance fails, the cluster and the Domain Administration Server (DAS) are aware of the failure and can take action when failure occurs. Many features of GlassFish Server depend upon GMS. For example, GMS is used by the in-memory session replication, transaction service, and timer service features.

If server instances in a cluster are located on different machines, ensure that all the server instance machines and the DAS machine are on the same subnet and that multicast is enabled for the network. To test whether multicast is enabled, use the `validate-multicast(1)` subcommand.

GMS is a core service of the Shoa! framework. For more information about Shoa!, visit the [Project Shoa! home page \(https://shoal.dev.java.net/\)](https://shoal.dev.java.net/).

The following topics are addressed here:

- “GMS Configuration Settings” on page 66
- “Dotted Names for GMS Settings” on page 68
- “To Preconfigure Nondefault GMS Configuration Settings” on page 69
- “To Change GMS Settings After Cluster Creation” on page 70
- “To Check the Health of Instances in a Cluster” on page 71
- “To Validate That Multicast Transport Is Available for a Cluster” on page 71
- “Using the Multi-Homing Feature With GMS” on page 73

GMS Configuration Settings

GlassFish Server has two types of GMS settings:

- *GMS cluster settings* — These are determined during cluster creation. For more information about these settings, see “To Create a Cluster” on page 75.
- *GMS configuration settings* — These are determined during configuration creation and are explained here.

The following *GMS configuration settings* are used in GMS for group discovery and failure detection:

`group-discovery-timeout-in-millis`

Indicates the amount of time (in milliseconds) an instance's GMS module will wait during instance startup for discovering other members of the group.

The `group-discovery-timeout-in-millis` timeout value should be set to the default or higher. The default is 5000.

`max-missed-heartbeats`

Indicates the maximum number of missed heartbeats that the health monitor counts before the instance can be marked as a suspected failure. GMS also tries to make a peer-to-peer connection with the suspected member. If the maximum number of missed heartbeats is exceeded and peer-to-peer connection fails, the member is marked as a suspected failure. The default is 3.

`heartbeat-frequency-in-millis`

Indicates the frequency (in milliseconds) at which a heartbeat is sent by each server instance to the cluster.

The failure detection interval is the `max-missed-heartbeats` multiplied by the `heartbeat-frequency-in-millis`. Therefore, the combination of defaults, 3 multiplied by 2000 milliseconds, results in a failure detection interval of 6 seconds.

Lowering the value of `heartbeat-frequency-in-millis` below the default would result in more frequent heartbeat messages being sent out from each member. This could potentially result in more heartbeat messages in the network than a system needs for triggering failure detection protocols. The effect of this varies depending on how quickly the deployment environment needs to have failure detection performed. That is, the (lower) number of retries with a lower heartbeat interval would make it quicker to detect failures.

However, lowering this value could result in false positives because you could potentially detect a member as failed when, in fact, the member's heartbeat is reflecting the network load from other parts of the server. Conversely, a higher timeout interval results in fewer heartbeats in the system because the time interval between heartbeats is longer. As a result, failure detection would take a longer. In addition, a startup by a failed member during this time results in a new join notification but no failure notification, because failure detection and verification were not completed.

The default is 2000.

`verify-failure-waittime-in-millis`

Indicates the verify suspect protocol's timeout used by the health monitor. After a member is marked as suspect based on missed heartbeats and a failed peer-to-peer connection check, the verify suspect protocol is activated and waits for the specified timeout to check for any

further health state messages received in that time, and to see if a peer-to-peer connection can be made with the suspect member. If not, then the member is marked as failed and a failure notification is sent. The default is 1500.

`verify-failure-connect-timeout-in-millis`

Indicates the time it takes for the GMS to detect a hardware or network failure of a server instance. Be careful not to set this value too low. The smaller this timeout value is, the greater the chance of detecting false failures. That is, the instance has not failed but doesn't respond within the short window of time. The default is 10000.

The heartbeat frequency, maximum missed heartbeats, peer-to-peer connection-based failure detection, and the verify timeouts are all needed to ensure that failure detection is robust and reliable in GlassFish Server.

For the dotted names for each of these GMS configuration settings, see [“Dotted Names for GMS Settings” on page 68](#). For the steps to specify these settings, see [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#).

Dotted Names for GMS Settings

Below are sample `get(1)` subcommands to get all the GMS configuration settings (attributes associated with the referenced `mycfg` configuration) and GMS cluster settings (attributes and properties associated with a cluster named `mycluster`).

```
asadmin> get "configs.config.mycfg.group-management-service.*"
configs.config.mycfg.group-management-service.failure-detection.heartbeat-frequency-in-millis=2000
configs.config.mycfg.group-management-service.failure-detection.max-missed-heartbeats=3
configs.config.mycfg.group-management-service.failure-detection.verify-failure-connect-timeout-in-millis=10000
configs.config.mycfg.group-management-service.failure-detection.verify-failure-waittime-in-millis=1500
configs.config.mycfg.group-management-service.group-discovery-timeout-in-millis=5000
```

```
asadmin> get clusters.cluster.mycluster
clusters.cluster.mycluster.config-ref=mycfg
clusters.cluster.mycluster.gms-bind-interface-address=${GMS-BIND-INTERFACE-ADDRESS-mycluster}
clusters.cluster.mycluster.gms-enabled=true
clusters.cluster.mycluster.gms-multicast-address=228.9.245.47
clusters.cluster.mycluster.gms-multicast-port=9833
clusters.cluster.mycluster.name=mycluster
```

```
asadmin> get "clusters.cluster.mycluster.property.*"
clusters.cluster.mycluster.property.GMS_LISTENER_PORT=${GMS_LISTENER_PORT-mycluster}
clusters.cluster.mycluster.property.GMS_MULTICAST_TIME_TO_LIVE=4
clusters.cluster.mycluster.property.GMS_LOOPBACK=false
clusters.cluster.mycluster.property.GMS_TCPSTARTPORT=9090
clusters.cluster.mycluster.property.GMS_TCPENDPORT=9200
```

The last `get` subcommand displays only the properties that have been explicitly set.

For the steps to specify these settings, see [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#) and [“To Change GMS Settings After Cluster Creation” on page 70](#).

▼ To Preconfigure Nondefault GMS Configuration Settings

You can preconfigure GMS with values different than the defaults without requiring a restart of the DAS and the cluster.

1 Create a configuration using the `copy-config(1)` subcommand.

For example:

```
asadmin> copy-config default-config mycfg
```

For more information, see [“To Create a Named Configuration” on page 119](#).

2 Set the values for the new configuration's GMS configuration settings.

For example:

```
asadmin > set configs.config.mycfg.group-management-service.group-discovery-timeout-in-millis=8000
asadmin> set configs.config.mycfg.group-management-service.failure-detection.max-missed-heartbeats=5
```

For a complete list of the dotted names for these settings, see [“Dotted Names for GMS Settings” on page 68](#).

3 Create the cluster so it uses the previously created configuration.

For example:

```
asadmin> create-cluster --config mycfg mycluster
```

You can also set GMS cluster settings during this step. For more information, see [“To Create a Cluster” on page 75](#).

4 Create server instances for the cluster.

For example:

```
asadmin> create-instance --node localhost --cluster mycluster instance01
```

```
asadmin> create-instance --node localhost --cluster mycluster instance02
```

5 Start the cluster.

For example:

```
asadmin> start-cluster mycluster
```

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

▼ To Change GMS Settings After Cluster Creation

To avoid the need to restart the DAS and the cluster, configure GMS configuration settings before cluster creation as explained in [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#).

To avoid the need to restart the DAS and the cluster, configure the GMS cluster settings during cluster creation as explained in [“To Create a Cluster” on page 75](#).

Changing any GMS settings using the set subcommand after cluster creation requires a domain administration server (DAS) and cluster restart as explained here.

1 Ensure that the DAS and cluster are running.

Remote subcommands require a running server.

2 Use the get(1) subcommand to determine the settings to change.

For example:

```
asadmin> get "configs.config.mycfg.group-management-service.*"
configs.config.mycfg.group-management-service.failure-detection.heartbeat-frequency-in-millis=2000
configs.config.mycfg.group-management-service.failure-detection.max-missed-heartbeats=3
configs.config.mycfg.group-management-service.failure-detection.verify-failure-connect-timeout-in-millis=10000
configs.config.mycfg.group-management-service.failure-detection.verify-failure-waittime-in-millis=1500
configs.config.mycfg.group-management-service.group-discovery-timeout-in-millis=5000
```

For a complete list of the dotted names for these settings, see [“Dotted Names for GMS Settings” on page 68](#).

3 Use the set(1) subcommand to change the settings.

For example:

```
asadmin> set configs.config.mycfg.group-management-service.group-discovery-timeout-in-millis=6000
```

4 Use the get subcommand again to confirm that the changes were made.

For example:

```
asadmin> get configs.config.mycfg.group-management-service.group-discovery-timeout-in-millis
```

5 Restart the DAS.

For example:

```
asadmin> stop-domain domain1
```

```
asadmin> start-domain domain1
```

6 Restart the cluster.

For example:

```
asadmin> stop-cluster mycluster
```

```
asadmin> start-cluster mycluster
```

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

▼ To Check the Health of Instances in a Cluster

The `get-health` subcommand only works when GMS is enabled. This is the quickest way to evaluate the health of a cluster and to detect if cluster is properly operating; that is, all members of the cluster are running and visible to DAS.

If multicast is not enabled for the network, all instances could be running (as shown by the `list-instances(1)` subcommand), yet isolated from each other. The `get-health` subcommand does not show the instances if they are running but cannot discover each other due to multicast not being configured properly. See [“To Validate That Multicast Transport Is Available for a Cluster” on page 71](#).

- 1 **Ensure that the DAS and cluster are running.**
Remote subcommands require a running server.
- 2 **Check whether server instances in a cluster are running by using the `get-health(1)` subcommand.**

Example 4–1 Checking the Health of Instances in a Cluster

This example checks the health of a cluster named `cluster1`.

```
asadmin> get-health cluster1
instance1 started since Wed Sep 29 16:32:46 EDT 2010
instance2 started since Wed Sep 29 16:32:45 EDT 2010
Command get-health executed successfully.
```

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

▼ To Validate That Multicast Transport Is Available for a Cluster

Before You Begin To test a specific multicast address, multicast port, or bind interface address, get this information beforehand using the `get` subcommand. Use the following subcommand to get the multicast address and port for a cluster named `c1`:

```
asadmin> get clusters.cluster.c1
clusters.cluster.c1.config-ref=mycfg
clusters.cluster.c1.gms-bind-interface-address=${GMS-BIND-INTERFACE-ADDRESS-c1}
```

```
clusters.cluster.c1.gms-enabled=true
clusters.cluster.c1.gms-multicast-address=228.9.174.162
clusters.cluster.c1.gms-multicast-port=5383
clusters.cluster.c1.name=c1
```

Use the following subcommand to get the bind interface address of a server instance named `i1` that belongs to a cluster named `c1`, if this system property has been set:

```
asadmin> get servers.server.i1.system-property.GMS-BIND-INTERFACE-ADDRESS-c1
servers.server.i1.system-property.GMS-BIND-INTERFACE-ADDRESS-c1.name=GMS-BIND-INTERFACE-ADDRESS-c1
servers.server.i1.system-property.GMS-BIND-INTERFACE-ADDRESS-c1.value=10.12.152.30
```

For information on how to set this system property, see [“Using the Multi-Homing Feature With GMS” on page 73](#).

Note – Do not run the `validate-multicast` subcommand using the DAS and cluster's multicast address and port values while the DAS and cluster are running. Doing so results in an error.

The `validate-multicast` subcommand must be run at the same time on two or more machines to validate whether multicast messages are being received between the machines.

- **Check whether multicast transport is available for a cluster by using the `validate-multicast(1)` subcommand.**

Example 4–2 Validating That Multicast Transport Is Available for a Cluster

This example checks whether multicast transport is available for a cluster named `c1`.

Run from host `sr1`:

```
asadmin> validate-multicast
Will use port 2048
Will use address 228.9.3.1
Will use bind interface null
Will use wait period 2,000 (in milliseconds)

Listening for data...
Sending message with content "sr1" every 2,000 milliseconds
Received data from sr1 (loopback)
Received data from sr2
Exiting after 20 seconds. To change this timeout, use the --timeout command line option.
Command validate-multicast executed successfully.
```

Run from host `sr2`:

```
asadmin> validate-multicast
Will use port 2048
Will use address 228.9.3.1
Will use bind interface null
Will use wait period 2,000 (in milliseconds)
```



```

Listening for data...
Sending message with content "sr2" every 2,000 milliseconds
Received data from sr2 (loopback)
Received data from sr1
Exiting after 20 seconds. To change this timeout, use the --timeout command line option.
Command validate-multicast executed successfully.

```

Next Steps As long as all machines see each other, multicast is validated to be working properly across the machines. If the machines are not seeing each other, set the `--bindaddress` option explicitly to ensure that all machines are using interface on same subnet, or increase the `--timetolive` option from the default of 4. If these changes fail to resolve the multicast issues, ask the network administrator to verify that the network is configured so the multicast messages can be seen between all the machines used to run the cluster.

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

Using the Multi-Homing Feature With GMS

Multi-homing enables GlassFish Server clusters to be used in an environment that uses multiple Network Interface Cards (NICs). A multi-homed host has multiple network connections, of which the connections may or may not be the on same network. Multi-homing provides the following benefits:

- Provides redundant network connections within the same subnet. Having multiple NICs ensures that one or more network connections are available for communication.
- Supports communication across two or more different subnets. The DAS and all server instances in the same cluster must be on the same subnet for GMS communication, however.
- Binds to a specific IPv4 address and receives GMS messages in a system that has multiple IP addresses configured. The responses for GMS messages received on a particular interface will also go out through that interface.
- Supports separation of external and internal traffic.

Traffic Separation Using Multi-Homing

You can separate the internal traffic resulting from GMS from the external traffic. Traffic separation enables you plan a network better and augment certain parts of the network, as required.

Consider a simple cluster, `c1`, with three instances, `i101`, `i102`, and `i103`. Each instance runs on a different machine. In order to separate the traffic, the multi-homed machine should have at least two IP addresses belonging to different networks. The first IP as the external IP and the

second one as internal IP. The objective is to expose the external IP to user requests, so that all the traffic from the user requests would be through them. The internal IP is used only by the cluster instances for internal communication through GMS. The following procedure describes how to set up traffic separation.

To configure multi-homed machines for GMS without traffic separation, skip the steps or commands that configure the `EXTERNAL-ADDR` system property, but perform the others.

To avoid having to restart the DAS or cluster, perform the following steps in the specified order.

▼ To Set Up Traffic Separation

1 Create the system properties `EXTERNAL-ADDR` and `GMS-BIND-INTERFACE-ADDRESS-c1` for the DAS.

- `asadmin create-system-properties --target server EXTERNAL-ADDR=192.155.35.4`
- `asadmin create-system-properties --target server
GMS-BIND-INTERFACE-ADDRESS-c1=10.12.152.20`

2 Create the cluster with the default settings.

Use the following command:

```
asadmin create-cluster c1
```

A reference to a system property for GMS traffic is already set up by default in the `gms-bind-interface-address` cluster setting. The default value of this setting is `${GMS-BIND-INTERFACE-ADDRESS-cluster-name}`.

3 When creating the clustered instances, configure the external and GMS IP addresses.

Use the following commands:

- `asadmin create-instance --node localhost --cluster c1 --systemproperties
EXTERNAL-ADDR=192.155.35.5:GMS-BIND-INTERFACE-ADDRESS-c1=10.12.152.30 i101`
- `asadmin create-instance --node localhost --cluster c1 --systemproperties
EXTERNAL-ADDR=192.155.35.6:GMS-BIND-INTERFACE-ADDRESS-c1=10.12.152.40 i102`
- `asadmin create-instance --node localhost --cluster c1 --systemproperties
EXTERNAL-ADDR=192.155.35.7:GMS-BIND-INTERFACE-ADDRESS-c1=10.12.152.50 i103`

4 Set the address attribute of HTTP listeners to refer to the `EXTERNAL-ADDR` system properties.

Use the following commands:

```
asadmin set c1-config.network-config.network-listeners.network-listener.http-1.address=\${EXTERNAL-ADDR}  
asadmin set c1-config.network-config.network-listeners.network-listener.http-2.address=\${EXTERNAL-ADDR}
```

Creating, Listing, and Deleting Clusters

GlassFish Server enables you to create clusters, obtain information about clusters, and delete clusters that are no longer required.

The following topics are addressed here:

- [“To Create a Cluster” on page 75](#)
- [“To List All Clusters in a Domain” on page 78](#)
- [“To Delete a Cluster” on page 78](#)

▼ To Create a Cluster

Use the `create-cluster` subcommand in remote mode to create a cluster.

To ensure that the GMS can detect changes in cluster membership, a cluster's GMS settings must be configured correctly. To avoid the need to restart the DAS and the cluster, configure a cluster's GMS settings when you create the cluster. If you change GMS settings for an existing cluster, the DAS and the cluster must be restarted to apply the changes.

When you create a cluster, GlassFish Server automatically creates a Message Queue cluster for the GlassFish Server cluster. For more information about Message Queue clusters, see [“Using Message Queue Broker Clusters With GlassFish Server” on page 151](#).

Before You Begin

If the cluster is to reference an existing named configuration, ensure that the configuration exists. For more information, see [“To Create a Named Configuration” on page 119](#). If you are using a named configuration to preconfigure GMS settings, ensure that these settings have the required values in the named configuration. For more information, see [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#).

If you are configuring the cluster's GMS settings when you create the cluster, ensure that you have the following information:

- The address on which GMS listens for group events
- The port number of the communication port on which GMS listens for group events
- The maximum number of iterations or transmissions that a multicast message for GMS events can experience before the message is discarded
- The lowest port number in the range of ports from which GMS selects a TCP port on which to listen
- The highest port number in the range of ports from which GMS selects a TCP port on which to listen

If the DAS is running on a multihomed host, ensure that you have the Internet Protocol (IP) address of the network interface on the DAS host to which GMS binds.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the create-cluster subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the cluster, see the `create-cluster(1)` help page.

```
asadmin> create-cluster --config configuration
--multicastaddress multicast-address --multicastport multicast-port
--properties GMS_MULTICAST_TIME_TO_LIVE=max-iterations:
GMS_TCPSTARTPORT=start-port:GMS_TCPENDPORT=end-port cluster-name
```

configuration

An existing named configuration that the cluster is to reference.

multicast-address

The address on which GMS listens for group events.

multicast-port

The port number of the communication port on which GMS listens for group events.

max-iterations

The maximum number of iterations or transmissions that a multicast message for GMS events can experience before the message is discarded.

start-port

The lowest port number in the range of ports from which GMS selects a TCP port on which to listen. The default is 9090.

end-port

The highest port number in the range of ports from which GMS selects a TCP port on which to listen. The default is 9200.

cluster-name

Your choice of name for the cluster that you are creating.

3 If necessary, create a system property to represent the IP address of the network interface on the DAS host to which GMS binds.

This step is necessary only if the DAS is running on a multihome host.

```
asadmin> create-system-properties
GMS-BIND-INTERFACE-ADDRESS-cluster-name=das-bind-address
```

cluster-name

The name that you assigned to the cluster in [Step 2](#).

das-bind-address

The IP address of the network interface on the DAS host to which GMS binds.

Example 4-3 Creating a Cluster

This example creates a cluster that is named `ltscluster` for which port 1169 is to be used for secure IIOP connections. Because the `--config` option is not specified, the cluster references a copy of the named configuration `default-config` that is named `ltscluster-config`.

```
asadmin> create-cluster
--systemproperties IIOP_SSL_LISTENER_PORT=1169
ltscluster
Command create-cluster executed successfully.
```

Example 4-4 Creating a Cluster and Setting GMS Options

This example creates a cluster that is named `pmdcluster`, which references the existing configuration `clusterpresets` and for which the cluster's GMS settings are configured as follows:

- GMS listens for group events on address 228.9.3.1 and port 2048.
- A multicast message for GMS events is discarded after 3 iterations or transmissions.
- GMS selects a TCP port on which to listen from ports in the range 10000–10100.

```
asadmin> create-cluster --config clusterpresets
--multicastaddress 228.9.3.1 --multicastport 2048
--properties GMS_MULTICAST_TIME_TO_LIVE=3:
GMS_TCPSTARTPORT=10000:GMS_TCPENDPORT=10100 pmdcluster
Command create-cluster executed successfully.
```

Next Steps After creating a cluster, you can add GlassFish Server instances to the cluster as explained in the following sections:

- [“To Create an Instance Centrally” on page 82](#)
- [“To Create an Instance Locally” on page 92](#)

- See Also**
- [“To Create a Named Configuration” on page 119](#)
 - [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#)
 - [“Using Message Queue Broker Clusters With GlassFish Server” on page 151](#)
 - `create-cluster(1)`
 - `create-system-properties(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help create-cluster`
- `asadmin help create-system-properties`

▼ To List All Clusters in a Domain

Use the `list-clusters` subcommand in remote mode to obtain information about existing clusters in a domain.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `list-clusters(1)` subcommand.

```
asadmin> list-clusters
```

Example 4–5 Listing All Clusters in a Domain

This example lists all clusters in the current domain.

```
asadmin> list-clusters
pmdclust not running
ymlclust not running
Command list-clusters executed successfully.
```

Example 4–6 Listing All Clusters That Are Associated With a Node

This example lists the clusters that contain an instance that resides on the node `sj01`.

```
asadmin> list-clusters sj01
ymlclust not running
Command list-clusters executed successfully.
```

See Also `list-clusters(1)`

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

▼ To Delete a Cluster

Use the `delete-cluster` subcommand in remote mode to remove a cluster from the DAS configuration.

If the cluster's named configuration was created automatically for the cluster and no other clusters or unclustered instances refer to the configuration, the configuration is deleted when the cluster is deleted.

Before You Begin Ensure that following prerequisites are met:

- The cluster that you are deleting is stopped. For information about how to stop a cluster, see [“To Stop a Cluster” on page 88](#).

- The cluster that you are deleting contains no GlassFish Server instances. For information about how to remove instances from a cluster, see the following sections:
 - [“To Delete an Instance Centrally” on page 86](#)
 - [“To Delete an Instance Locally” on page 96](#)

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Confirm that the cluster is stopped.

```
asadmin> list-clusters cluster-name
cluster-name
```

The name of the cluster that you are deleting.

3 Confirm that the cluster contains no instances.

```
asadmin> list-instances cluster-name
cluster-name
```

The name of the cluster that you are deleting.

4 Run the delete-cluster(1) subcommand.

```
asadmin> delete-cluster cluster-name
cluster-name
```

The name of the cluster that you are deleting.

Example 4-7 Deleting a Cluster

This example confirms that the cluster `adcccluster` is stopped and contains no instances and deletes the cluster `adcccluster`.

```
asadmin> list-clusters adcccluster
adcccluster not running
Command list-clusters executed successfully.
asadmin> list-instances adcccluster
Nothing to list.
Command list-instances executed successfully.
asadmin> delete-cluster adcccluster
Command delete-cluster executed successfully.
```

- See Also**
- [“To Stop a Cluster” on page 88](#)
 - [“To Delete an Instance Centrally” on page 86](#)
 - [“To Delete an Instance Locally” on page 96](#)
 - `delete-cluster(1)`
 - `list-clusters(1)`
 - `list-instances(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-cluster`
- `asadmin help list-clusters`
- `asadmin help list-instances`

Administering GlassFish Server Instances

A GlassFish Server *instance* is a single Virtual Machine for the Java platform (Java Virtual Machine or JVM machine) on a single node in which GlassFish Server is running. A node defines the host where the GlassFish Server instance resides. The JVM machine must be compatible with the Java Platform, Enterprise Edition (Java EE).

GlassFish Server instances form the basis of an application deployment. An instance is a building block in the clustering, load balancing, and session persistence features of GlassFish Server. Each instance belongs to a single domain and has its own directory structure, configuration, and deployed applications. Every instance contains a reference to a node that defines the host where the instance resides.

The following topics are addressed here:

- [“Types of GlassFish Server Instances” on page 81](#)
- [“Administering GlassFish Server Instances Centrally” on page 82](#)
- [“Administering GlassFish Server Instances Locally” on page 92](#)
- [“Resynchronizing GlassFish Server Instances and the DAS” on page 101](#)
- [“Migrating EJB Timers” on page 114](#)

Types of GlassFish Server Instances

Each GlassFish Server instance is one of the following types of instance:

Standalone instance

A standalone instance does not share its configuration with any other instances or clusters. A standalone instance is created if either of the following conditions is met:

- No configuration or cluster is specified in the command to create the instance.
- A configuration that is not referenced by any other instances or clusters is specified in the command to create the instance.

When no configuration or cluster is specified, a copy of the `default-config` configuration is created for the instance. The name of this configuration is *instance-name-config*, where *instance-name* represents the name of an unclustered server instance.

Shared instance

A shared instance shares its configuration with other instances or clusters. A shared instance is created if a configuration that is referenced by other instances or clusters is specified in the command to create the instance.

Clustered instance

A clustered instance inherits its configuration from the cluster to which the instance belongs and shares its configuration with other instances in the cluster. A clustered instance is created if a cluster is specified in the command to create the instance.

Any instance that is not part of a cluster is considered an unclustered server instance. Therefore, standalone instances and shared instances are unclustered server instances.

Administering GlassFish Server Instances Centrally

Centralized administration requires secure shell (SSH) to be set up. If SSH is set up, you can administer clustered instances without the need to log in to hosts where remote instances reside.

Administering GlassFish Server instances centrally involves the following tasks:

- [“To Create an Instance Centrally” on page 82](#)
- [“To List All Instances in a Domain” on page 85](#)
- [“To Delete an Instance Centrally” on page 86](#)
- [“To Start a Cluster” on page 87](#)
- [“To Stop a Cluster” on page 88](#)
- [“To Start an Individual Instance Centrally” on page 89](#)
- [“To Stop an Individual Instance Centrally” on page 90](#)
- [“To Restart an Individual Instance Centrally” on page 91](#)

▼ To Create an Instance Centrally

Use the `create-instance` subcommand in remote mode to create a GlassFish Server instance centrally. Creating an instance adds the instance to the DAS configuration and creates the instance's files on the host where the instance resides.

If the instance is a clustered instance that is managed by GMS, system properties for the instance that relate to GMS must be configured correctly. To avoid the need to restart the DAS and the instance, configure an instance's system properties that relate to GMS when you create

the instance. If you change GMS-related system properties for an existing instance, the DAS and the instance must be restarted to apply the changes. For information about GMS, see [“Group Management Service” on page 66](#).

Before You Begin Ensure that following prerequisites are met:

- The node where the instance is to reside exists.
- The node where the instance is to reside is either enabled for remote communication or represents the host on which the DAS is running. For information about how to create a node that is enabled for remote communication, see [“To Create an SSH Node” on page 52](#).
- The user of the DAS can use SSH to log in to the host for the node where the instance is to reside.

If any of these prerequisites is not met, create the instance locally as explained in [“To Create an Instance Locally” on page 92](#).

If you are adding the instance to a cluster, ensure that the cluster to which you are adding the instance exists. For information about how to create a cluster, see [“To Create a Cluster” on page 75](#).

If the instance is to reference an existing named configuration, ensure that the configuration exists. For more information, see [“To Create a Named Configuration” on page 119](#).

The instance might be a clustered instance that is managed by GMS and resides on a node that represents a multihome host. In this situation, ensure that you have the Internet Protocol (IP) address of the network interface to which GMS binds.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `create-instance` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the instance, see the `create-instance(1)` help page.

- **If you are creating a standalone instance, do not specify a cluster.**

If the instance is to reference an existing configuration, specify a configuration that no other cluster or instance references.

```
asadmin> create-instance --node node-name
[ --config configuration-name] instance-name
```

node-name

The node on which the instance is to reside.

configuration-name

The name of the existing named configuration that the instance will reference.

If you do not require the instance to reference an existing configuration, omit this option. A copy of the `default-config` configuration is created for the instance. The name of this configuration is *instance-name-config*, where *instance-name* is the name of the server instance.

instance-name

Your choice of name for the instance that you are creating.

- **If you are creating a shared instance, specify the configuration that the instance will share with other clusters or instances.**

Do *not* specify a cluster.

```
asadmin> create-instance --node node-name  
--config configuration-name instance-name
```

node-name

The node on which the instance is to reside.

configuration-name

The name of the existing named configuration that the instance will reference.

instance-name

Your choice of name for the instance that you are creating.

- **If you are creating a clustered instance, specify the cluster to which the instance will belong.**

If the instance is managed by GMS and resides on a node that represents a multihome host, specify the `GMS-BIND-INTERFACE-ADDRESS-cluster-name` system property.

```
asadmin> create-instance --cluster cluster-name --node node-name  
[--systemproperties GMS-BIND-INTERFACE-ADDRESS-cluster-name=bind-address] instance-name
```

cluster-name

The name of the cluster to which you are adding the instance.

node-name

The node on which the instance is to reside.

bind-address

The IP address of the network interface to which GMS binds. Specify this option only if the instance is managed by GMS and resides on a node that represents a multihome host.

instance-name

Your choice of name for the instance that you are creating.

Example 5–1 Creating a Clustered Instance Centrally

This example adds the instance `pmd-i1` to the cluster `pmdclust` in the domain `domain1`. The instance resides on the node `sj01`, which represents the host `sj01.example.com`.

```

asadmin> create-instance --cluster pmdclust --node sj01 pmd-i1
Port Assignments for server instance pmd-i1:
JMX_SYSTEM_CONNECTOR_PORT=28686
JMS_PROVIDER_PORT=27676
HTTP_LISTENER_PORT=28080
ASADMIN_LISTENER_PORT=24848
IIOP_SSL_LISTENER_PORT=23820
IIOP_LISTENER_PORT=23700
HTTP_SSL_LISTENER_PORT=28181
IIOP_SSL_MUTUALAUTH_PORT=23920
The instance, pmd-i1, was created on host sj01.example.com
Command create-instance executed successfully.

```

- See Also**
- [“To Create an SSH Node” on page 52](#)
 - [“To Create an Instance Locally” on page 92](#)
 - `create-instance(1)`

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

Next Steps After creating an instance, you can start the instance as explained in the following sections:

- [“To Start an Individual Instance Centrally” on page 89](#)
- [“To Stop an Individual Instance Locally” on page 99](#)

▼ To List All Instances in a Domain

Use the `list-instances` subcommand in remote mode to obtain information about existing instances in a domain.

- 1 **Ensure that the DAS is running.**
Remote subcommands require a running server.
- 2 **Run the `list-instances(1)` subcommand.**

```
asadmin> list-instances
```

Example 5–2 Listing Basic Information About All GlassFish Server Instances in a Domain

This example lists the name and status of all GlassFish Server instances in the current domain.

```

asadmin> list-instances
pmd-i2 running
yml-i2 running
pmd-i1 running
yml-i1 running
pmdsa1 not running
Command list-instances executed successfully.

```

Example 5–3 Listing Detailed Information About All GlassFish Server Instances in a Domain

This example lists detailed information about all GlassFish Server instances in the current domain.

```
asadmin> list-instances --long=true
NAME      HOST                PORT    PID    CLUSTER    STATE
pmd-i1    sj01.example.com    24848   31310   pmdcluster running
yml-i1    sj01.example.com    24849   25355   ymlcluster running
pmdsa1    localhost           24848   -1      ---        not running
pmd-i2    sj02.example.com    24848   22498   pmdcluster running
yml-i2    sj02.example.com    24849   20476   ymlcluster running
ymlsa1    localhost           24849   -1      ---        not running
Command list-instances executed successfully.
```

See Also `list-instances(1)`

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

▼ To Delete an Instance Centrally

Use the `delete-instance` subcommand in remote mode to delete a GlassFish Server instance centrally.



Caution – If you are using a Java Message Service (JMS) cluster with a master broker, do not delete the instance that is associated with the master broker. If this instance must be deleted, use the `change-master-broker(1)` subcommand to assign the master broker to a different instance.

Deleting an instance involves the following:

- Removing the instance from the configuration of the DAS
- Deleting the instance's files from file system

Before You Begin Ensure that the instance that you are deleting is not running. For information about how to stop an instance, see the following sections:

- [“To Stop an Individual Instance Centrally” on page 90](#)
- [“To Stop an Individual Instance Locally” on page 99](#)

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Confirm that the instance is not running.

```
asadmin> list-instances instance-name
```

instance-name

The name of the instance that you are deleting.

3 Run the `delete-instance(1)` subcommand.

```
asadmin> delete-instance instance-name
```

instance-name

The name of the instance that you are deleting.

Example 5–4 Deleting an Instance Centrally

This example confirms that the instance `pmd-i1` is not running and deletes the instance.

```
asadmin> list-instances pmd-i1
pmd-i1    not running
Command list-instances executed successfully.
asadmin> delete-instance pmd-i1
Command _delete-instance-filesystem executed successfully.
The instance, pmd-i1, was deleted from host sj01.example.com
Command delete-instance executed successfully.
```

- See Also**
- [“To Stop an Individual Instance Centrally” on page 90](#)
 - [“To Stop an Individual Instance Locally” on page 99](#)
 - `change-master-broker(1)`
 - `delete-instance(1)`
 - `list-instances(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-instance`
- `asadmin help list-instances`

▼ To Start a Cluster

Use the `start-cluster` subcommand in remote mode to start a cluster.

Starting a cluster starts all instances in the cluster that are not already running.

Before You Begin Ensure that following prerequisites are met:

- Each node where an instance in the cluster resides is either enabled for remote communication or represents the host on which the DAS is running.
- The user of the DAS can use secure shell (SSH) to log in to the host for any node where instances in the cluster reside.

If any of these prerequisites is not met, start the cluster by starting each instance locally as explained in [“To Start an Individual Instance Locally” on page 98](#).

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `start-cluster(1)` subcommand.

```
asadmin> start-cluster cluster-name
```

cluster-name

The name of the cluster that you are starting.

Example 5-5 Starting a Cluster

This example starts the cluster `pmdcluster`.

```
asadmin> start-cluster pmdcluster  
Command start-cluster executed successfully.
```

- See Also**
- [“To Start an Individual Instance Locally” on page 98](#)
 - `start-cluster(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help start-cluster` at the command line.

Next Steps After starting a cluster, you can deploy applications to the cluster. For more information, see *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

▼ To Stop a Cluster

Use the `stop-cluster` subcommand in remote mode to stop a cluster.

Stopping a cluster stops all running instances in the cluster.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `stop-cluster(1)` subcommand.

```
asadmin> stop-cluster cluster-name
```

cluster-name

The name of the cluster that you are stopping.

Example 5–6 Stopping a Cluster

This example stops the cluster `pmdcluster`.

```
asadmin> stop-cluster pmdcluster
Command stop-cluster executed successfully.
```

See Also `stop-cluster(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help stop-cluster` at the command line.

Troubleshooting If instances in the cluster have become unresponsive and fail to stop, run the subcommand again with the `--kill` option set to `true`. When this option is `true`, the subcommand uses functionality of the operating system to kill the process for each running instance in the cluster.

▼ To Start an Individual Instance Centrally

Use the `start-instance` subcommand in remote mode to start an individual instance centrally.

Before You Begin Ensure that following prerequisites are met:

- The node where the instance resides is either enabled for remote communication or represents the host on which the DAS is running.
- The user of the DAS can use SSH to log in to the host for the node where the instance resides.

If any of these prerequisites is not met, start the instance locally as explained in [“To Start an Individual Instance Locally” on page 98](#).

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `start-instance` subcommand.

```
asadmin> start-instance instance-name
```

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-instance(1)` help page.

instance-name

The name of the instance that you are starting.

Example 5-7 Starting an Individual Instance Centrally

This example starts the instance `pmd-i2`, which resides on the node `sj02`. This node represents the host `sj02.example.com`. The configuration of the instance on this node already matched the configuration of the instance in the DAS when the instance was started.

```
asadmin> start-instance pmd-i2
CLI801 Instance is already synchronized
Waiting for pmd-i2 to start .....
Successfully started the instance: pmd-i2
Instance Location: /export/glassfish3/glassfish/nodes/sj02/pmd-i2
Log File: /export/glassfish3/glassfish/nodes/sj02/pmd-i2/logs/server.log
Admin Port: 24851
Command start-local-instance executed successfully.
The instance, pmd-i2, was started on host sj02.example.com
Command start-instance executed successfully.
```

See Also `start-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help start-instance` at the command line.

Next Steps After starting an instance, you can deploy applications to the instance. For more information, see *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

▼ To Stop an Individual Instance Centrally

Use the `stop-instance` subcommand in remote mode to stop an individual instance centrally.

When an instance is stopped, the instance stops accepting new requests and waits for all outstanding requests to be completed.

- 1 **Ensure that the DAS is running.**
Remote subcommands require a running server.
- 2 **Run the `stop-instance(1)` subcommand.**

Example 5-8 Stopping an Individual Instance Centrally

This example stops the instance `pmd-i2`.

```
asadmin> stop-instance pmd-i2
The instance, pmd-i2, is stopped.
Command stop-instance executed successfully.
```

See Also `stop-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help stop-instance` at the command line.

Troubleshooting If the instance has become unresponsive and fails to stop, run the subcommand again with the `--kill` option set to `true`. When this option is `true`, the subcommand uses functionality of the operating system to kill the instance process.

▼ To Restart an Individual Instance Centrally

Use the `restart-instance` subcommand in remote mode to start an individual instance centrally.

When this subcommand restarts an instance, the DAS synchronizes the instance with changes since the last synchronization as described in [“Default Synchronization for Files and Directories” on page 101](#).

If you require different synchronization behavior, stop and start the instance as explained in [“To Resynchronize an Instance and the DAS Online” on page 104](#).

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `restart-instance(1)` subcommand.

```
asadmin> restart-instance instance-name
instance-name
```

The name of the instance that you are restarting.

Example 5–9 Restarting an Individual Instance Centrally

This example restarts the instance `pmd-i2`.

```
asadmin> restart-instance pmd-i2
pmd-i2 was restarted.
Command restart-instance executed successfully.
```

See Also

- [“To Stop an Individual Instance Centrally” on page 90](#)
- [“To Start an Individual Instance Centrally” on page 89](#)
- `restart-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help restart-instance` at the command line.

Troubleshooting If the instance has become unresponsive and fails to stop, run the subcommand again with the `--kill` option set to `true`. When this option is `true`, the subcommand uses functionality of the operating system to kill the instance process before restarting the instance.

Administering GlassFish Server Instances Locally

Local administration does not require SSH to be set up. If SSH is not set up, you must log in to each host where remote instances reside and administer the instances individually.

Administering GlassFish Server instances locally involves the following tasks:

- [“To Create an Instance Locally” on page 92](#)
- [“To Delete an Instance Locally” on page 96](#)
- [“To Start an Individual Instance Locally” on page 98](#)
- [“To Stop an Individual Instance Locally” on page 99](#)
- [“To Restart an Individual Instance Locally” on page 100](#)

Note – Even if SSH is not set up, you can obtain information about instances in a domain without logging in to each host where remote instances reside. For instructions, see [“To List All Instances in a Domain” on page 85](#).

▼ To Create an Instance Locally

Use the `create-local-instance` subcommand in remote mode to create a GlassFish Server instance locally. Creating an instance adds the instance to the DAS configuration and creates the instance's files on the host where the instance resides.

If the instance is a clustered instance that is managed by GMS, system properties for the instance that relate to GMS must be configured correctly. To avoid the need to restart the DAS and the instance, configure an instance's system properties that relate to GMS when you create the instance. If you change GMS-related system properties for an existing instance, the DAS and the instance must be restarted to apply the changes. For information about GMS, see [“Group Management Service” on page 66](#).

Before You Begin If you plan to specify the node on which the instance is to reside, ensure that the node exists.

Note – If you create the instance on a host for which no nodes are defined, you can create the instance without creating a node beforehand. In this situation, GlassFish Server creates a `CONFIG` node for you. The name of the node is the unqualified name of the host.

For information about how to create a node, see the following sections:

- [“To Create an SSH Node” on page 52](#)
- [“To Create a CONFIG Node” on page 56](#)

If you are adding the instance to a cluster, ensure that the cluster to which you are adding the instance exists. For information about how to create a cluster, see [“To Create a Cluster” on page 75](#).

If the instance is to reference an existing named configuration, ensure that the configuration exists. For more information, see [“To Create a Named Configuration” on page 119](#).

The instance might be a clustered instance that is managed by GMS and resides on a node that represents a multihome host. In this situation, ensure that you have the Internet Protocol (IP) address of the network interface to which GMS binds.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Log in to the host that is represented by the node where the instance is to reside.

3 Run the `create-local-instance` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the instance, see the `create-local-instance(1)` help page.

- **If you are creating a standalone instance, do not specify a cluster.**

If the instance is to reference an existing configuration, specify a configuration that no other cluster or instance references.

```
$ asadmin --host das-host [--port admin-port]
create-local-instance [--node node-name] [--config configuration-name] instance-name
das-host
```

das-host
The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

node-name

The node on which the instance is to reside.

If you are creating the instance on a host for which fewer than two nodes are defined, you may omit this option.

- If no nodes are defined for the host, GlassFish Server creates a CONFIG node for you. The name of the node is the unqualified name of the host.
- If one node is defined for the host, the instance is created on that node.

configuration-name

The name of the existing named configuration that the instance will reference.

If you do not require the instance to reference an existing configuration, omit this option. A copy of the `default-config` configuration is created for the instance. The name of this configuration is *instance-name-config*, where *instance-name* is the name of the server instance.

instance-name

Your choice of name for the instance that you are creating.

- **If you are creating a shared instance, specify the configuration that the instance will share with other clusters or instances.**

Do *not* specify a cluster.

```
$ asadmin --host das-host [--port admin-port]
create-local-instance [--node node-name] --config configuration-name instance-name
```

das-host

The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

node-name

The node on which the instance is to reside.

If you are creating the instance on a host for which fewer than two nodes are defined, you may omit this option.

- If no nodes are defined for the host, GlassFish Server creates a CONFIG node for you. The name of the node is the unqualified name of the host.
- If one node is defined for the host, the instance is created on that node.

configuration-name

The name of the existing named configuration that the instance will reference.

instance-name

Your choice of name for the instance that you are creating.

- **If you are creating a clustered instance, specify the cluster to which the instance will belong.**

If the instance is managed by GMS and resides on a node that represents a multihome host, specify the `GMS-BIND-INTERFACE-ADDRESS-cluster-name` system property.

```
$ asadmin --host das-host [--port admin-port]
create-local-instance --cluster cluster-name [--node node-name]
[--systemproperties GMS-BIND-INTERFACE-ADDRESS-cluster-name=bind-address] instance-name
```

das-host

The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

cluster-name

The name of the cluster to which you are adding the instance.

node-name

The node on which the instance is to reside.

If you are creating the instance on a host for which fewer than two nodes are defined, you may omit this option.

- If no nodes are defined for the host, GlassFish Server creates a CONFIG node for you. The name of the node is the unqualified name of the host.
- If one node is defined for the host, the instance is created on that node.

bind-address

The IP address of the network interface to which GMS binds. Specify this option only if the instance is managed by GMS and resides on a node that represents a multihome host.

instance-name

Your choice of name for the instance that you are creating.

Example 5–10 Creating a Clustered Instance Locally Without Specifying a Node

This example adds the instance `kui-i1` to the cluster `kuicluster` locally. The CONFIG node `xk01` is created automatically to represent the host `xk01.example.com`, on which this example is run. The DAS is running on the host `dashost.example.com` and listens for administration requests on the default port.

The commands to list the nodes in the domain are included in this example only to demonstrate the creation of the node `xk01`. These commands are not required to create the instance.

```
$ asadmin --host dashost.example.com list-nodes --long
NODE NAME          TYPE      NODE HOST          INSTALL DIRECTORY  REFERENCED BY
localhost-domain1  CONFIG    localhost          /export/glassfish3
Command list-nodes executed successfully.
$ asadmin --host dashost.example.com
create-local-instance --cluster kuicluster kui-i1
Rendezvoused with DAS on dashost.example.com:4848.
Port Assignments for server instance kui-i1:
JMX_SYSTEM_CONNECTOR_PORT=28687
JMS_PROVIDER_PORT=27677
HTTP_LISTENER_PORT=28081
ASADMIN_LISTENER_PORT=24849
JAVA_DEBUGGER_PORT=29009
IIOP_SSL_LISTENER_PORT=23820
IIOP_LISTENER_PORT=23700
OSGI_SHELL_TELNET_PORT=26666
HTTP_SSL_LISTENER_PORT=28182
```

```
IIOP_SSL_MUTUALAUTH_PORT=23920
Command create-local-instance executed successfully.
$ asadmin --host dashost.example.com list-nodes --long
NODE NAME          TYPE    NODE HOST          INSTALL DIRECTORY  REFERENCED BY
localhost-domain1  CONFIG localhost         /export/glassfish3
xk01               CONFIG xk01.example.com  /export/glassfish3  kui-i1
Command list-nodes executed successfully.
```

Example 5–11 Creating a Clustered Instance Locally

This example adds the instance `yml-i1` to the cluster `ymlcluster` locally. The instance resides on the node `sj01`. The DAS is running on the host `das1.example.com` and listens for administration requests on the default port.

```
$ asadmin --host das1.example.com
create-local-instance --cluster ymlcluster --node sj01 yml-i1
Rendezvoused with DAS on das1.example.com:4848.
Port Assignments for server instance yml-i1:
JMX_SYSTEM_CONNECTOR_PORT=28687
JMS_PROVIDER_PORT=27677
HTTP_LISTENER_PORT=28081
ASADMIN_LISTENER_PORT=24849
JAVA_DEBUGGER_PORT=29009
IIOP_SSL_LISTENER_PORT=23820
IIOP_LISTENER_PORT=23700
OSGI_SHELL_TELNET_PORT=26666
HTTP_SSL_LISTENER_PORT=28182
IIOP_SSL_MUTUALAUTH_PORT=23920
Command create-local-instance executed successfully.
```

- See Also**
- [“To Create an SSH Node” on page 52](#)
 - [“To Create a CONFIG Node” on page 56](#)
 - `create-local-instance(1)`

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

Next Steps After creating an instance, you can start the instance as explained in the following sections:

- [“To Start an Individual Instance Centrally” on page 89](#)
- [“To Stop an Individual Instance Locally” on page 99](#)

▼ To Delete an Instance Locally

Use the `delete-local-instance` subcommand in remote mode to delete a GlassFish Server instance locally.



Caution – If you are using a Java Message Service (JMS) cluster with a master broker, do not delete the instance that is associated with the master broker. If this instance must be deleted, use the `change-master-broker(1)` subcommand to assign the master broker to a different instance.

Deleting an instance involves the following:

- Removing the instance from the configuration of the DAS
- Deleting the instance's files from file system

Before You Begin Ensure that the instance that you are deleting is not running. For information about how to stop an instance, see the following sections:

- [“To Stop an Individual Instance Centrally” on page 90](#)
- [“To Stop an Individual Instance Locally” on page 99](#)

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Log in to the host that is represented by the node where the instance resides.

3 Confirm that the instance is not running.

```
$ asadmin --host das-host [--port admin-port]
list-instances instance-name
```

das-host

The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

instance-name

The name of the instance that you are deleting.

4 Run the `delete-local-instance(1)` subcommand.

```
$ asadmin --host das-host [--port admin-port]
delete-local-instance [--node node-name] instance-name
```

das-host

The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

node-name

The node on which the instance resides. If only one node is defined for the GlassFish Server installation that you are running on the node's host, you may omit this option.

instance-name

The name of the instance that you are deleting.

Example 5–12 Deleting an Instance Locally

This example confirms that the instance `yml-il` is not running and deletes the instance.

```
$ asadmin --host das1.example.com list-instances yml-il
yml-il    not running
Command list-instances executed successfully.
$ asadmin --host das1.example.com delete-local-instance --node sj01 yml-il
Command delete-local-instance executed successfully.
```

- See Also**
- [“To Stop an Individual Instance Centrally” on page 90](#)
 - [“To Stop an Individual Instance Locally” on page 99](#)
 - `change-master-broker(1)`
 - `delete-local-instance(1)`
 - `list-instances(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-local-instance`
- `asadmin help list-instances`

▼ To Start an Individual Instance Locally

Use the `start-local-instance` subcommand in local mode to start an individual instance locally.

- 1 **Log in to the host that is represented by the node where the instance resides.**
- 2 **Run the `start-local-instance` subcommand.**

```
$ asadmin start-local-instance [--node node-name] instance-name
```

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-local-instance(1)` help page.

node-name

The node on which the instance resides. If only one node is defined for the GlassFish Server installation that you are running on the node's host, you may omit this option.

instance-name

The name of the instance that you are starting.

Example 5–13 Starting an Individual Instance Locally

This example starts the instance `yml-i1` locally. The instance resides on the node `sj01`.

```
$ asadmin start-local-instance --node sj01 yml-i1
Waiting for yml-i1 to start .....
Successfully started the instance: yml-i1
Instance Location: /export/glassfish3/glassfish/nodes/sj01/yml-i1
Log File: /export/glassfish3/glassfish/nodes/sj01/yml-i1/logs/server.log
Admin Port: 24849
Command start-local-instance executed successfully.
```

See Also `start-local-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help start-local-instance` at the command line.

Next Steps After starting an instance, you can deploy applications to the instance. For more information, see *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

▼ To Stop an Individual Instance Locally

Use the `stop-local-instance` subcommand in local mode to stop an individual instance locally.

When an instance is stopped, the instance stops accepting new requests and waits for all outstanding requests to be completed.

1 Log in to the host that is represented by the node where the instance resides.

2 Run the `stop-local-instance(1)` subcommand.

```
$ asadmin stop-local-instance [--node node-name] instance-name
```

node-name

The node on which the instance resides. If only one node is defined for the GlassFish Server installation that you are running on the node's host, you may omit this option.

instance-name

The name of the instance that you are stopping.

Example 5–14 Stopping an Individual Instance Locally

This example stops the instance `ym1-i1` locally. The instance resides on the node `sj01`.

```
$ asadmin stop-local-instance --node sj01 yml-i1
Waiting for the instance to stop ....
Command stop-local-instance executed successfully.
```

See Also `stop-local-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help stop-local-instance` at the command line.

Troubleshooting If the instance has become unresponsive and fails to stop, run the subcommand again with the `--kill` option set to `true`. When this option is `true`, the subcommand uses functionality of the operating system to kill the instance process.

▼ To Restart an Individual Instance Locally

Use the `restart-local-instance` subcommand in local mode to restart an individual instance locally.

When this subcommand restarts an instance, the DAS synchronizes the instance with changes since the last synchronization as described in [“Default Synchronization for Files and Directories” on page 101](#).

If you require different synchronization behavior, stop and start the instance as explained in [“To Resynchronize an Instance and the DAS Online” on page 104](#).

1 Log in to the host that is represented by the node where the instance resides.

2 Run the `restart-local-instance` subcommand.

```
$ asadmin restart-local-instance [--node node-name] instance-name
node-name
```

The node on which the instance resides. If only one node is defined for the GlassFish Server installation that you are running on the node's host, you may omit this option.

```
instance-name
```

The name of the instance that you are restarting.

Example 5–15 Restarting an Individual Instance Locally

This example restarts the instance `ym1-i1` locally. The instance resides on the node `sj01`.

```
$ asadmin restart-local-instance --node sj01 yml-i1
Command restart-local-instance executed successfully.
```

See Also `restart-local-instance(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help restart-local-instance` at the command line.

Troubleshooting If the instance has become unresponsive and fails to stop, run the subcommand again with the `--kill` option set to `true`. When this option is `true`, the subcommand uses functionality of the operating system to kill the instance process before restarting the instance.

Resynchronizing GlassFish Server Instances and the DAS

Configuration data for a GlassFish Server instance is stored as follows:

- In the repository of the domain administration server (DAS)
- In a cache on the host that is local to the instance

The configuration data in these locations must be synchronized. The cache is synchronized in the following circumstances:

- Whenever an `asadmin` subcommand is run. For more information, see “Impact of Configuration Changes” in *GlassFish Server Open Source Edition 3.1 Administration Guide*.
- When a user uses the administration tools to start or restart an instance.

Default Synchronization for Files and Directories

The `--sync` option of the subcommands for starting an instance controls the type of synchronization between the DAS and the instance's files when the instance is started. You can use this option to override the default synchronization behavior for the files and directories of an instance. For more information, see [“To Resynchronize an Instance and the DAS Online” on page 104](#).

On the DAS, the files and directories of an instance are stored in the *domain-dir* directory, where *domain-dir* is the directory in which a domain's configuration is stored. The default synchronization behavior for the files and directories of an instance is as follows:

`applications`

This directory contains a subdirectory for each application that is deployed to the instance.

By default, only a change to an application's top-level directory within the application directory causes the DAS to synchronize that application's directory. When the DAS resynchronizes the `applications` directory, all the application's files and all generated content that is related to the application are copied to the instance.

If a file below a top-level subdirectory is changed without a change to a file in the top-level subdirectory, full synchronization is required. In normal operation, files below the top-level

subdirectories of these directories are not changed and such files should not be changed by users. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

`config`

This directory contains configuration files for the entire domain.

By default, the DAS resynchronizes files that have been modified since the last resynchronization only if the `domain.xml` file in this directory has been modified.

Note – If you add a file to the `config` directory of an instance, the file is deleted when the instance is resynchronized with the DAS. However, any file that you add to the `config` directory of the DAS is *not* deleted when instances and the DAS are resynchronized. By default, any file that you add to the `config` directory of the DAS is not resynchronized. If you require any additional configuration files to be resynchronized, you must specify the files explicitly. For more information, see [“To Resynchronize Additional Configuration Files” on page 110](#).

`config`

`config/config-name`

This directory contains files that are to be shared by all instances that reference the named configuration *config-name*. A *config-name* directory exists for each named configuration in the configuration of the DAS.

Because the *config-name* directory contains the subdirectories `lib` and `docroot`, this directory might be very large. Therefore, by default, only a change to a file or a top-level subdirectory of *config-name* causes the DAS to resynchronize the *config-name* directory.

`config/domain.xml`

This file contains the DAS configuration for the domain to which the instance belongs.

By default, the DAS resynchronizes this file if it has been modified since the last resynchronization.

Note – A change to the `config/domain.xml` file is required to cause the DAS to resynchronize an instance's files. If the `config/domain.xml` file has not changed since the last resynchronization, none of the instance's files is resynchronized, even if some of these files are out of date in the cache.

`docroot`

This directory is the HTTP document root directory. By default, all instances in a domain use the same document root directory. To enable instances to use a different document root directory, a virtual server must be created in which the `docroot` property is set. For more information, see the `create-virtual-server(1)` help page.

The docroot directory might be very large. Therefore, by default, only a change to a file or a subdirectory in the top level of the docroot directory causes the DAS to resynchronize the docroot directory. The DAS checks files in the top level of the docroot directory to ensure that changes to the `index.html` file are detected.

When the DAS resynchronizes the docroot directory, all modified files and subdirectories at any level are copied to the instance.

If a file below a top-level subdirectory is changed without a change to a file in the top-level subdirectory, full synchronization is required.

`generated`

This directory contains generated files for Java EE applications and modules, for example, EJB stubs, compiled JSP classes, and security policy files. Do not modify the contents of this directory.

This directory is resynchronized when the `applications` directory is resynchronized. Therefore, only directories for applications that are deployed to the instance are resynchronized.

`java-web-start`

This directory is *not* resynchronized. It is created and populated as required on each instance.

`lib`

`lib/classes`

These directories contain common Java class files or JAR archives and ZIP archives for use by applications that are deployed to the entire domain. Typically, these directories contain common JDBC drivers and other utility libraries that are shared by all applications in the domain.

The contents of these directories are loaded by the common class loader. For more information, see “Using the Common Class Loader” in *GlassFish Server Open Source Edition 3.1 Application Development Guide*. The class loader loads the contents of these directories in the following order:

1. `lib/classes`
2. `lib/*.jar`
3. `lib/*.zip`

The `lib` directory also contains the following subdirectories:

`applibs`

This directory contains application-specific Java class files or JAR archives and ZIP archives for use by applications that are deployed to the entire domain.

ext

This directory contains optional packages in JAR archives and ZIP archives for use by applications that are deployed to the entire domain. These archive files are loaded by using Java extension mechanism. For more information, see [Optional Packages - An Overview](#).

Note – Optional packages were formerly known as *standard extensions* or *extensions*.

The `lib` directory and its subdirectories typically contain only a small number of files. Therefore, by default, a change to any file in these directories causes the DAS to resynchronize the file that changed.

▼ To Resynchronize an Instance and the DAS Online

Resynchronizing an instance and the DAS updates the instance with changes to the instance's configuration files on the DAS. An instance is resynchronized with the DAS when the instance is started or restarted.

Note – Resynchronization of an instance is only required if the instance is stopped. A running instance does *not* require resynchronization.

- 1 **Ensure that the DAS is running.**
- 2 **Determine whether the instance is stopped.**

```
asadmin> list-instances instance-name  
instance-name
```

The name of the instance that you are resynchronizing with the DAS.

If the instance is stopped, the `list-instances` subcommand indicates that the instance is not running.

- 3 **If the instance is stopped, start the instance.**

If the instance is running, no further action is required.

- **If SSH is set up, start the instance centrally.**

If you require full synchronization, set the `--sync full` option of the `start-instance` subcommand to `full`. If default synchronization is sufficient, omit this option.

```
asadmin> start-instance [--sync full] instance-name
```

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-instance(1)` help page.

instance-name

The name of the instance that you are starting.

- **If SSH is not set up, start the instance locally from the host where the instance resides.**

If you require full synchronization, set the `--sync` option of the `start-local-instance` subcommand to `full`. If default synchronization is sufficient, omit this option.

```
$ asadmin start-local-instance [--node node-name] [--sync full] instance-name
```

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-local-instance(1)` help page.

node-name

The node on which the instance resides. If only one node is defined for the GlassFish Server installation that you are running on the node's host, you may omit this option.

instance-name

The name of the instance that you are starting.

Example 5–16 Resynchronizing an Instance and the DAS Online

This example determines that the instance `yml-i1` is stopped and fully resynchronizes the instance with the DAS. Because SSH is not set up, the instance is started locally on the host where the instance resides. In this example, multiple nodes are defined for the GlassFish Server installation that is running on the node's host.

To determine whether the instance is stopped, the following command is run in multimode on the DAS host:

```
asadmin> list-instances yml-i1
yml-i1    not running
Command list-instances executed successfully.
```

To start the instance, the following command is run in single mode on the host where the instance resides:

```
$ asadmin start-local-instance --node sj01 --sync full yml-i1
Removing all cached state for instance yml-i1.
Waiting for yml-i1 to start .....
```

```
Successfully started the instance: yml-i1
instance Location: /export/glassfish3/glassfish/nodes/sj01/yml-i1
Log File: /export/glassfish3/glassfish/nodes/sj01/yml-i1/logs/server.log
Admin Port: 24849
Command start-local-instance executed successfully.
```

- See Also**
- `list-instances(1)`
 - `start-instance(1)`
 - `start-local-instance(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help list-instances
```

```
asadmin help start-instance
```

```
asadmin help start-local-instance
```

▼ To Resynchronize Library Files

To ensure that library files are resynchronized correctly, you must ensure that each library file is placed in the correct directory for the type of file. You must add files in some directories for library files to the class path yourself. Files in other directories are added to the class path automatically.

- 1 Place each library file in the correct location for the type of library file as shown in the following table.**

Type of Library Files	Location
Common JAR archives and ZIP archives for all applications in a domain.	<i>domain-dir/lib</i>
Common Java class files for a domain for all applications in a domain.	<i>domain-dir/lib/classes</i>
Application-specific libraries.	<i>domain-dir/lib/applibs</i>
Optional packages for all applications in a domain.	<i>domain-dir/lib/ext</i>
Library files for all applications that are deployed to a specific cluster or standalone instance.	<i>domain-dir/config/config-name/lib</i>
Optional packages for all applications that are deployed to a specific cluster or standalone instance.	<i>domain-dir/config/config-name/lib/ext</i>

domain-dir

The directory in which the domain's configuration is stored.

config-name

For a standalone instance: the named configuration that the instance references.

For a clustered instance: the named configuration that the cluster to which the instance belongs references.

- 2 **For library files for all applications that are deployed to a specific cluster or standalone instance, add the files to the class path prefix or class path suffix of the named configuration that the cluster or standalone instance references.**

This type of library file is stored in the *domain-dir/config/config-name/lib* directory.

For other types of library file, omit this step. Other types of library file are automatically added to the class path by their class loaders.

- **To add the files to the class path prefix, perform these steps:**

- a. **Determine the existing class path prefix.**

```
asadmin> get configs.config.config-name.java-config.classpath-prefix
```

config-name

The named configuration that the cluster or standalone instance references.

- b. **Set the class path prefix to the existing class path prefix and the files that you are adding.**

```
asadmin> set
configs.config.config-name.java-config.classpath-prefix=library-file-list
```

config-name

The named configuration that the cluster or standalone instance references.

library-file-list

A comma-separated list of the files in the existing class path prefix and the files that you are adding.

- **To add the files to the class path suffix, perform these steps:**

- a. **Determine the existing class path suffix.**

```
asadmin> get configs.config.config-name.java-config.classpath-suffix
```

config-name

The named configuration that the cluster or standalone instance references.

- b. **Set the class path suffix to the existing class path suffix and the files that you are adding.**

```
asadmin> set
configs.config.config-name.java-config.classpath-suffix=library-file-list
```

config-name

The named configuration that the cluster or standalone instance references.

library-file-list

A comma-separated list of the files in the existing class path suffix and the files that you are adding.

Next Steps When you deploy an application that depends on these library files, use the `--libraries` option of the `deploy` subcommand to specify these dependencies. For library files in the *domain-dir/lib/applib* directory, only the JAR file name is required, for example:

```
asadmin deploy --libraries commons-coll.jar,X1.jar app.ear
```

For other types of library file, the full path is required.

See Also

- `get(1)`
- `set(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help get
```

```
asadmin help set
```

▼ To Resynchronize Custom Configuration Files for an Instance

Configuration files in the *domain-dir/config* directory that are resynchronized are resynchronized for the entire domain. If you create a custom configuration file for an instance or a cluster, the custom file is resynchronized only for the instance or cluster.

- 1 **Place the custom configuration file in the *domain-dir/config/config-name* directory.**

domain-dir

The directory in which the domain's configuration is stored.

config-name

The named configuration that the instance references.

- 2 **If the instance locates the file through an option of the Java application launcher, update the option.**

- a. **Delete the option.**

```
asadmin> delete-jvm-options --target instance-name  
option-name=current-value
```

instance-name

The name of the instance for which the custom configuration file is created.

option-name

The name of the option for locating the file.

current-value

The current value of the option for locating the file.

b. Re-create the option that you deleted in the previous step.

```
asadmin> create-jvm-options --target instance-name
option-name=new-value
```

instance-name

The name of the instance for which the custom configuration file is created.

option-name

The name of the option for locating the file.

new-value

The new value of the option for locating the file.

Example 5–17 Updating the Option for Locating a Configuration File

This example updates the option for locating the `server.policy` file to specify a custom file for the instance `pmd`.

```
asadmin> delete-jvm-options --target pmd
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
Deleted 1 option(s)
Command delete-jvm-options executed successfully.
asadmin> create-jvm-options --target pmd
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/pmd-config/server.policy
Created 1 option(s)
Command create-jvm-options executed successfully.
```

- See Also**
- `create-jvm-options(1)`
 - `delete-jvm-options(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help create-jvm-options
```

```
asadmin help delete-jvm-options
```

▼ To Resynchronize Users' Changes to Files

A change to the `config/domain.xml` file is required to cause the DAS to resynchronize instances' files. If other files in the domain directory are changed without a change to the `config/domain.xml` file, instances are not resynchronized with these changes.

The following changes are examples of changes to the domain directory without a change to the `config/domain.xml` file:

- Adding files to the `lib` directory
- Adding certificates to the key store by using the `keytool` command

1 Change the last modified time of the `config/domain.xml` file.

Exactly how to change the last modified time depends on the operating system. For example, on UNIX and Linux systems, you can use the `touch(1)` command.

2 Resynchronize each instance in the domain with the DAS.

For instructions, see [“To Resynchronize an Instance and the DAS Online” on page 104](#).

- See Also**
- [“To Resynchronize an Instance and the DAS Online” on page 104](#)
 - `touch(1)`

▼ To Resynchronize Additional Configuration Files

By default, GlassFish Server synchronizes only the following configuration files:

- `admin-keyfile`
- `cacerts.jks`
- `default-web.xml`
- `domain.xml`
- `domain-passwords`
- `keyfile`
- `keystore.jks`
- `server.policy`
- `sun-acc.xml`
- `wss-server-config-1.0`
- `xml wss-server-config-2.0.xml`

If you require instances in a domain to be resynchronized with additional configuration files for the domain, you can specify a list of files to resynchronize.



Caution – If you specify a list of files to resynchronize, you must specify *all* the files that the instances require, including the files that GlassFish Server resynchronizes by default. Any file in the instance's cache that is not in the list is deleted when the instance is resynchronized with the DAS.

- **In the `config` directory of the domain, create a plain text file that is named `config-files` that lists the files to resynchronize.**

In the `config-files` file, list each file name on a separate line.

Example 5–18 `config-files` File

This example shows the content of a `config-files` file. This file specifies that the `some-other-info` file is to be resynchronized in addition to the files that GlassFish Server resynchronizes by default:

```
admin-keyfile
cacerts.jks
default-web.xml
domain.xml
domain-passwords
keyfile
keystore.jks
server.policy
sun-acc.xml
wss-server-config-1.0.xml
wss-server-config-2.0.xml
some-other-info
```

▼ To Prevent Deletion of Application-Generated Files

When the DAS resynchronizes an instance's files, the DAS deletes from the instance's cache any files that are not listed for resynchronization. If an application creates files in a directory that the DAS resynchronizes, these files are deleted when the DAS resynchronizes an instance with the DAS.

- **Put the files in a subdirectory under the domain directory that is not defined by GlassFish Server, for example, `/export/glassfish3/glassfish/domains/domain1/myapp/myfile`.**

▼ To Resynchronize an Instance and the DAS Offline

Resynchronizing an instance and the DAS offline updates the instance's cache without the need for the instance to be able to communicate with the DAS. Offline resynchronization is typically required for the following reasons:

- To reestablish the instance after an upgrade

- To synchronize the instance manually with the DAS when the instance cannot contact the DAS

- 1 **Ensure that the DAS is running.**
- 2 **Export the configuration data that you are resynchronizing to an archive file.**

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for exporting the configuration data, see the `export-sync-bundle(1)` help page.

How to export the data depends on the host from where you run the `export-sync-bundle` subcommand.

- **From the DAS host, run the `export-sync-bundle` subcommand as follows:**

```
asadmin> export-sync-bundle --target target
target
```

The cluster or standalone instance for which to export configuration data.

Do *not* specify a clustered instance. If you specify a clustered instance, an error occurs. To export configuration data for a clustered instance, specify the name of the cluster of which the instance is a member, not the instance.

The file is created on the DAS host.

- **From the host where the instance resides, run the `export-sync-bundle` subcommand as follows:**

```
$ asadmin --host das-host [--port admin-port]
export-sync-bundle [--retrieve=true] --target target
das-host
```

The name of the host where the DAS is running.

admin-port

The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

target

The cluster or standalone instance for which to export configuration data.

Do *not* specify a clustered instance. If you specify a clustered instance, an error occurs. To export configuration data for a clustered instance, specify the name of the cluster of which the instance is a member, not the instance.

Note – To create the archive file on the host where the instance resides, set the `--retrieve` option to `true`. If you omit this option, the archive file is created on the DAS host.

- 3 If necessary, copy the archive file that you created in [Step 2](#) from the DAS host to the host where the instance resides.
 - 4 From the host where the instance resides, import the instance's configuration data from the archive file that you created in [Step 2](#).
-

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for importing the configuration data, see the `import-sync-bundle(1)` help page.

```
$ asadmin import-sync-bundle [--node node-name] --instance instance-name archive-file
node-name
```

The node on which the instance resides. If you omit this option, the subcommand determines the node from the DAS configuration in the archive file.

instance-name

The name of the instance that you are resynchronizing.

archive-file

The name of the file, including the path, that contains the archive file to import.

Example 5–19 Resynchronizing an Instance and the DAS Offline

This example resynchronizes the clustered instance `ym1-i1` and the DAS offline. The instance is a member of the cluster `ym1cluster`. The archive file that contains the instance's configuration data is created on the host where the instance resides.

```
$ asadmin --host dashost.example.com
export-sync-bundle --retrieve=true --target ymlcluster
Command export-sync-bundle executed successfully.
$ asadmin import-sync-bundle --node sj01
--instance yml-i1 ymlcluster-sync-bundle.zip
Command import-sync-bundle executed successfully.
```

- See Also**
- `export-sync-bundle(1)`
 - `import-sync-bundle(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help export-sync-bundle
```

```
asadmin help import-sync-bundle
```

Migrating EJB Timers

If a GlassFish Server server instance stops or fails abnormally, it may be desirable to migrate the EJB timers defined for that stopped server instance to another running server instance.

Automatic timer migration is enabled by default for clustered server instances that are stopped normally. Automatic timer migration can also be enabled to handle clustered server instance crashes. In addition, timers can be migrated manually for stopped or crashed server instances.

- [“To Enable Automatic EJB Timer Migration for Failed Clustered Instances” on page 114](#)
- [“To Migrate EJB Timers Manually” on page 114](#)

▼ To Enable Automatic EJB Timer Migration for Failed Clustered Instances

Automatic migration of EJB timers is enabled by default for clustered server instances that are stopped normally. If the Group Management Service (GMS) is enabled and a clustered instance is stopped normally, no further action is required for timer migration to occur. The procedure in this section is only necessary if you want to enable automatic timer migration for clustered server instances that have stopped abnormally.

Note – If the GMS is enabled, the default automatic timer migration cannot be disabled. To disable automatic timer migration, you must first disable the GMS. For information about the GMS, see [“Group Management Service” on page 66](#).

Before You Begin Automatic EJB timer migration can only be configured for clustered server instances. Automatic timer migration is not possible for standalone server instances.

- **Enable delegated transaction recovery for the cluster.**

This enables automatic timer migration for failed server instances in the cluster.

For instructions on enabling delegated transaction recovery, see Chapter 18, “Administering Transactions,” in *GlassFish Server Open Source Edition 3.1 Administration Guide*.

▼ To Migrate EJB Timers Manually

EJB timers can be migrated manually from a stopped source instance to a specified target instance in the same cluster if GMS notification is not enabled. If no target instance is specified, the DAS will attempt to find a suitable server instance. A migration notification will then be sent to the selected target server instance.

Note the following restrictions:

- If the source instance is part of a cluster, then the target instance must also be part of that same cluster.
- It is not possible to migrate timers from a standalone instance to a clustered instance, or from one cluster to another cluster.
- It is not possible to migrate timers from one standalone instance to another standalone instance.
- All EJB timers defined for a given instance are migrated with this procedure. It is not possible to migrate individual timers.

Before You Begin The server instance from which the EJB timers are to be migrated should not be active during the migration process.

- 1 **Verify that the source clustered server instance from which the EJB timers are to be migrated is not currently running.**

```
asadmin> list-instances source-instance
```

- 2 **Stop the instance from which the timers are to be migrated, if that instance is still running.**

```
asadmin> stop-instance source-instance
```

Note – The target instance to which the timers will be migrated should be running.

- 3 **(Optional) List the currently defined EJB timers on the source instance, if desired.**

```
asadmin> list-timers source-cluster
```

- 4 **Migrate the timers from the stopped source instance to the target instance.**

```
asadmin> migrate-timers --target target-instance source-instance
```

Example 5–20 Migrating an EJB Timer

The following example show how to migrate timers from a clustered source instance named `football` to a clustered target instance named `soccer`.

```
asadmin> migrate-timers --target soccer football
```

See Also `list-timers(1)`, `migrate-timers(1)`, `list-instances(1)`, `stop-instance(1)`

Administering Named Configurations

A *named configuration* is a set of configuration information for GlassFish Server instances and clusters. A configuration sets port numbers for listeners that an instance or a cluster uses and defines settings for items such as the EJB container, security, logging, and monitoring. Applications and resources are *not* defined in named configurations.

GlassFish Server enables you to create configurations for use by clusters and instances. GlassFish Server creates configurations automatically for clusters and instances that do not reference an existing configuration. You can modify any existing named configuration regardless of how the configuration was created.

The following topics are addressed here:

- [“About Named Configurations” on page 117](#)
- [“Creating, Listing, and Deleting Named Configurations” on page 119](#)
- [“Modifying Properties for Named Configurations and Instances” on page 123](#)

About Named Configurations

Configurations exist in a domain. Multiple GlassFish Server instances or clusters in the domain can reference the same configuration, or they can have separate configurations. To ensure that the environment in a cluster’s instances is homogenous, all instances in the cluster inherit the cluster’s configuration.

Types of Named Configurations

Each named configuration is one of the following types of configuration:

Standalone configuration

A standalone configuration is referenced by only one instance or cluster and is not shared with any other instances or clusters.

Note – A configuration that is referenced by only one cluster is a standalone configuration, even if the cluster contains multiple instances.

Shared configuration

A shared configuration is referenced by multiple instances or clusters.

The type of an unclustered instance is determined by the type of the configuration that the instance references. For more information, see [“Types of GlassFish Server Instances” on page 81](#).

The default-config Configuration

The default-config configuration is a special configuration that acts as a template for creating named configurations. Clusters and instances cannot refer to the default-config configuration. The default-config configuration can only be copied to create configurations.

Automatically Created Configurations

When you create a cluster or an instance, you can choose whether to specify an existing configuration that the new cluster or instance will reference. If you choose to create a cluster or an instance without specifying an existing configuration, GlassFish Server automatically creates a configuration for the cluster or instance. For more information, see the following sections:

- [“To Create a Cluster” on page 75](#)
- [“To Create an Instance Centrally” on page 82](#)
- [“To Create an Instance Locally” on page 92](#)

GlassFish Server automatically creates a configuration by copying the default-config configuration. If you require an instance or cluster to reference a copy of a different configuration, copy the configuration and specify the copy when you create the instance or cluster. For information about how to copy a configuration, see [“To Create a Named Configuration” on page 119](#).

GlassFish Server assigns the name *cluster-or-instance-config* to an automatically created configuration. *cluster-or-instance* is the name of the cluster or instance for which the configuration is created. The server-config configuration is automatically created for the domain administration server (DAS) when the domain is created.

Directory for Configuration Synchronization

When a named configuration is created, GlassFish Server creates a configuration directory on the domain administration server (DAS) at *domain-dir/config/config-name*.

domain-dir

The directory in which the domain's configuration is stored.

config-name

The name that was assigned to the configuration when the configuration was created.

This contents of this directory are synchronized to all instances that inherit or reference the configuration.

Creating, Listing, and Deleting Named Configurations

GlassFish Server enables you to create configurations for use by clusters and instances, obtain information about configurations, and delete configurations that are no longer required.

The following topics are addressed here:

- [“To Create a Named Configuration” on page 119](#)
- [“To List the Named Configurations in a Domain” on page 120](#)
- [“To List the Targets of a Named Configuration” on page 121](#)
- [“To Delete a Named Configuration” on page 122](#)

▼ To Create a Named Configuration

Use the `copy-config` subcommand in remote mode to create a named configuration by copying an existing configuration.

GlassFish Server requires you to create a configuration by copying a configuration because a configuration contains many required settings. The newly created configuration is identical to the configuration that you copy until you change its configuration settings.

Tip – Create a named configuration *only* if you plan to share the configuration among multiple unclustered instances or clusters, or if you are using a configuration to preconfigure GMS settings. Otherwise, create clusters and instances without specifying an existing configuration. If no configuration is specified, GlassFish Server creates a copy of the default configuration for the cluster or instance.

For more information, see the following sections:

- [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#)
 - [“To Create a Cluster” on page 75](#)
 - [“To Create an Instance Centrally” on page 82](#)
 - [“To Create an Instance Locally” on page 92](#)
-

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `copy-config` subcommand.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for configuring the named configuration, see the `copy-config(1)` help page.

`asadmin> copy-config source-config-name destination-config-name`

source-config-name

The name of the configuration that you are copying. You must specify a configuration to copy even if you are copying the default configuration. The configuration must already exist.

destination-config-name

Your choice of name for the configuration that you are creating by copying the source configuration.

Example 6–1 Creating a Named Configuration

This example creates the named configuration `clusterpresets-config` by copying the default configuration.

```
asadmin> copy-config default-config clusterpresets-config
Command copy-config executed successfully.
```

- See Also**
- [“To Preconfigure Nondefault GMS Configuration Settings” on page 69](#)
 - [“To Create a Cluster” on page 75](#)
 - [“To Create an Instance Centrally” on page 82](#)
 - [“To Create an Instance Locally” on page 92](#)
 - `copy-config(1)`

You can also view the full syntax and options of the subcommand by typing `asadmin help copy-config` at the command line.

▼ To List the Named Configurations in a Domain

Use the `list-configs` subcommand in remote mode to list existing named configurations in a domain.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Run the `list-configs(1)` subcommand.

```
asadmin> list-configs
```


Example 6-2 Listing All Named Configurations in a Domain

This example lists all named configurations in the current domain.

```
asadmin> list-configs
server-config
default-config
yamlcluster-config
clusterpresets-config
Command list-configs executed successfully.
```

See Also `list-configs(1)`

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

▼ To List the Targets of a Named Configuration

Use the `list-clusters` subcommand and the `list-instances` subcommand in remote mode to list the targets of a named configuration.

The targets of a named configuration are the clusters and GlassFish Server instances that reference the configuration.

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 List the clusters that refer to the configuration.

```
asadmin> list-clusters config-name
config-name
The name of the configuration whose targets you are listing.
```

3 List the instances that refer to the configuration.

```
asadmin> list-instances config-name
config-name
The name of the configuration whose targets you are listing.
```

Example 6-3 Listing the Targets of a Named Configuration

This example shows that the cluster `yamlcluster` and the instances `yaml-i1` and `yaml-i2` reference the named configuration `yamlcluster-config`.

```
asadmin> list-clusters yamlcluster-config
yamlcluster partially running
Command list-clusters executed successfully.
```

```
asadmin> list-instances ymlcluster-config
yml-i1    running
yml-i2    not running
Command list-instances executed successfully.
```

- See Also**
- `list-clusters(1)`
 - `list-instances(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help list-clusters`
- `asadmin help list-instances`

▼ To Delete a Named Configuration

Use the `delete-config` subcommand in remote mode to delete an existing named configuration from the configuration of the DAS.

You cannot delete the `default-config` configuration.

Note – A standalone configuration that was created automatically for a cluster or a GlassFish Server instance is deleted when the cluster or instance is deleted.

Before You Begin Ensure that no clusters or instances refer to the configuration. If a cluster or instance refers to the configuration and is no longer required, delete the cluster or instance. For information about how to delete an instance and how to delete a cluster, see the following sections:

- [“To Delete an Instance Centrally” on page 86](#)
- [“To Delete an Instance Locally” on page 96](#)
- [“To Delete a Cluster” on page 78](#)

1 Ensure that the DAS is running.

Remote subcommands require a running server.

2 Confirm that no clusters refer to the configuration that you are deleting.

```
asadmin> list-clusters config-name
```

```
config-name
```

The name of the configuration that you are deleting.

3 Confirm that no instances refer to the configuration that you are deleting.

```
asadmin> list-instances config-name
```

config-name

The name of the configuration that you are deleting.

4 Run the **delete-config(1)** subcommand.

```
asadmin> delete-config config-name
```

config-name

The name of the configuration that you are deleting.

Example 6–4 Deleting a Named Configuration

This example confirms that no clusters or instances refer to the configuration `clusterpresets-config` and then deletes the configuration.

```
asadmin> list-clusters clusterpresets-config
Nothing to list
Command list-clusters executed successfully.
asadmin> list-instances clusterpresets-config
Nothing to list.
Command list-instances executed successfully.
asadmin> delete-config clusterpresets-config
Command delete-config executed successfully.
```

- See Also**
- [“To Delete an Instance Centrally” on page 86](#)
 - [“To Delete an Instance Locally” on page 96](#)
 - [“To Delete a Cluster” on page 78](#)
 - `delete-config(1)`
 - `list-clusters(1)`
 - `list-instances(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help delete-config`
- `asadmin help list-clusters`
- `asadmin help list-instances`

Modifying Properties for Named Configurations and Instances

The properties in a named configuration define port numbers for unclustered instances that reference the configuration or clustered instances that inherit the configuration. An instance initially obtains port numbers from the configuration that the instance references or inherits. To avoid port conflicts, edit the properties of named configurations and instances.

The following topics are addressed here:

- [“Properties for Port Numbers in a Named Configuration” on page 124](#)
- [“To Modify a Named Configuration’s Properties” on page 125](#)
- [“To Modify Port Numbers of an Instance” on page 126](#)

Properties for Port Numbers in a Named Configuration

The default configuration `default-config` contains properties that define the initial values of port numbers in a configuration that is copied from `default-config`. When an instance or a cluster that references the configuration is created, these properties are set for the instance.

You can create additional system properties for a configuration either by specifying the `--systemproperties` option of the `copy-config(1)` subcommand or by using the `create-system-properties(1)` subcommand. To reference a system property from the configuration, use the `${prop-name}` notation, where *prop-name* is the name of the system property.

For example, if a configuration defines additional HTTP listeners, use system properties to define the ports for those listeners. However, these properties are not set automatically when an instance or a cluster that references the configuration is created. You must set these properties explicitly when you create the instance or cluster.

The properties in a named configuration that define port numbers are as follows:

ASADMIN_LISTENER_PORT

This property specifies the port number of the HTTP port or HTTPS port through which the DAS connects to the instance to manage the instance. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

HTTP_LISTENER_PORT

This property specifies the port number of the port that is used to listen for HTTP requests. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

HTTP_SSL_LISTENER_PORT

This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

IIOP_LISTENER_PORT

This property specifies the port number of the port that is used for IIOP connections. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

IIOP_SSL_LISTENER_PORT

This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

IIOP_SSL_MUTUALAUTH_PORT

This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

JAVA_DEBUGGER_PORT

This property specifies the port number of the port that is used for connections to the [Java Platform Debugger Architecture \(JPDA\)](#) debugger. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

JMS_PROVIDER_PORT

This property specifies the port number for the Java Message Service provider. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

JMX_SYSTEM_CONNECTOR_PORT

This property specifies the port number on which the JMX connector listens. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

OSGI_SHELL_TELNET_PORT

This property specifies the port number of the port that is used for connections to the [Apache Felix Remote Shell](#). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

▼ To Modify a Named Configuration's Properties

Use the `get` subcommand and the `set` subcommand in remote mode to modify a named configuration's properties.

You might copy a configuration for use by instances that reside on the same host as instances that refer to the original configuration. In this situation, edit the properties of one of the configurations to ensure that instances that will refer to the configuration have the correct initial settings.

If you change the port number in a configuration, the port number is changed for any instance that references or inherits the configuration.

- 1 Ensure that the DAS is running.**

Remote subcommands require a running server.

- 2 For each property that you are modifying, determine the current value and set the new value.**

- a. Determine the current value of the property.**

```
asadmin> get configs.config.config-name.system-property.property-name.value
```

config-name

The name of the configuration whose properties you are modifying.

property-name

The name of the property that you are modifying. For a list of available properties, see [“Properties for Port Numbers in a Named Configuration” on page 124.](#)

b. Set the property to its new value.

```
asadmin> set  
configs.config.config-name.system-property.property-name.value=new-value
```

config-name

The name of the configuration whose properties you are modifying.

property-name

The name of the property that you are modifying. For a list of available properties, see [“Properties for Port Numbers in a Named Configuration” on page 124.](#)

new-value

The value to which you are setting the property.

Example 6–5 Modifying a Property of a Named Configuration

This example changes the value of the `JMS_PROVIDER_PORT` property in the `clusterpresets-config` configuration from 27676 to 27678.

```
asadmin> get  
configs.config.clusterpresets-config.system-property.JMS_PROVIDER_PORT.value  
configs.config.clusterpresets-config.system-property.JMS_PROVIDER_PORT.value=27676  
Command get executed successfully.  
asadmin> set  
configs.config.clusterpresets-config.system-property.JMS_PROVIDER_PORT.value=27678  
configs.config.clusterpresets-config.system-property.JMS_PROVIDER_PORT.value=27678  
Command set executed successfully.
```

See Also

- `get(1)`
- `set(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help get`
- `asadmin help set`

▼ To Modify Port Numbers of an Instance

Use the `get` subcommand and the `set` subcommand in remote mode to modify the port numbers of an instance.

The port numbers of an instance are initially set in the configuration that the instance references or inherits from its parent cluster. Multiple instances that reside on the same host must each listen on a unique port number. Therefore, if multiple instances that reference or inherit the same configuration reside on the same host, a port conflict prevents all except one of the instances from starting. To avoid port conflicts, modify the port numbers on which individual instances listen.

If you modify an instance's port number and later modify the port number in the instance's configuration, the instance's port number remains unchanged.

The port numbers of an instance are stored as Java system properties. When GlassFish Server is started, it treats these properties in the same way as properties that are passed through the `-D` option of the Java application launcher.

- 1 Ensure that the DAS is running.**

Remote subcommands require a running server.

- 2 For each port number that you are modifying, determine the current value and set the new value.**

- a. Determine the current value of the port number.**

```
asadmin> get
servers.server.instance-name.system-property.port-property.value
instance-name
```

The name of the instance whose port numbers you are modifying.

port-property

The name of the property that corresponds to the port number that you are modifying.

For a list of available properties, see [“Properties for Port Numbers in a Named Configuration” on page 124](#).

- b. Set the port number to its new value.**

```
asadmin> get
servers.server.instance-name.system-property.port-property.value=new-value
instance-name
```

The name of the instance whose port numbers you are modifying.

port-property

The name of the property that corresponds to the port number that you are modifying.

For a list of available properties, see [“Properties for Port Numbers in a Named Configuration” on page 124](#).

new-value

The value to which you are setting the port number.

Example 6-6 **Modifying a Port Number for an Instance**

This example changes the port number of the HTTP port or the HTTPS port for administration of the `pmdsainst` instance from 24849 to 24859.

```
asadmin> get
servers.server.pmdsainst.system-property.ASADMIN_LISTENER_PORT.value
servers.server.pmdsainst.system-property.ASADMIN_LISTENER_PORT.value=24849
Command get executed successfully.
asadmin> set
servers.server.pmdsainst.system-property.ASADMIN_LISTENER_PORT.value=24859
servers.server.pmdsainst.system-property.ASADMIN_LISTENER_PORT.value=24859
Command set executed successfully.
```

- See Also**
- `get(1)`
 - `set(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line:

- `asadmin help get`
- `asadmin help set`

Configuring HTTP Load Balancing

This chapter describes how to configure HTTP load balancing on GlassFish Server 3.1.

The following topics are addressed here:

- [“Setting Up HTTP Load Balancing” on page 129](#)

For information on other types of load balancing, see [Chapter 10, “Configuring Java Message Service High Availability,”](#) and [Chapter 11, “RMI-IIOP Load Balancing and Failover.”](#)

Setting Up HTTP Load Balancing

This section describes how to set up load balancing for GlassFish Server.

The following topics are addressed here:

- [“Prerequisites for Setting Up HTTP Load Balancing” on page 129](#)
- [“Configuring GlassFish Server with Apache HTTP Server and mod_jk” on page 130](#)
- [“HTTP Load Balancer Deployments” on page 132](#)

Prerequisites for Setting Up HTTP Load Balancing

Before configuring your load balancer, you must:

- Install a supported web server and configure it. If using the mod_jk module, the only supported web server is Apache HTTP Server 2.2.x.
- Configure the mod_jk connector module, as described in [“Configuring GlassFish Server with Apache HTTP Server and mod_jk” on page 130.](#)
- Create GlassFish Server clusters or server instances to participate in load balancing.
- Deploy applications to these clusters or instances.

Configuring GlassFish Server with Apache HTTP Server and mod_jk

GlassFish Server 3.1 can be configured for load balancing with Apache HTTP Server as a front end by enabling the Apache mod_jk connector module. To enable the mod_jk module in GlassFish Server, set the GlassFish Server `jk-enablednetwork-listener` attribute. You can also create jk-connectors under different virtual-servers using the `jk-enablednetwork-listener` attribute.

▼ To Configure the mod_jk Connector Module

- 1 Install [Apache HTTP Server](#) and `mod_jk`.
- 2 Configure `workers.properties` and `httpd.conf`.

For example:

- `apache2/config/workers.properties`

```
# Define 1 real worker using ajp13
worker.list=worker1
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```
- `apache2/conf/httpd.conf`

```
LoadModule jk_module /Users/Amy/apache2/modules/mod_jk-1.2.25-httpd-2.2.4.so
JkWorkersFile /Users/Amy/apache2/conf/worker.properties
# Where to put jk logs
JkLogFile /Users/Amy/apache2/logs/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel debug
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"
# Send everything for context /examples to worker named worker1 (ajp13)
JkMount /examples/* worker1
```

- 3 Start Apache HTTP Server.
- 4 Enable mod_jk using the following commands.

Be sure to enter each of these commands on a single line.

```
asadmin> create-network-listener --protocol http-listener-1 \
--listenerport 8009 --jkenabled true jk-connector

asadmin> set server-config.network-config.network-listeners.network-listener.\
jk-connector.jk-configuration-file=domain-dir/config/glassfish-jk.properties
```

- 5 If you are using the `glassfish-jk.properties` file and not referencing it in `httpd.conf`, point to it using the following command:

```
asadmin create-jvm-options -Dcom.sun.enterprise.web.connector.enableJK.\
propertyFile=domain-dir/config/glassfish-jk.properties
```

- 6 Restart GlassFish Server.

Example 7-1 `httpd.conf` File for Load Balancing

This example shows an `httpd.conf` file that is set for load balancing.

```
LoadModule jk_module /usr/lib/httpd/modules/mod_jk.so
JkWorkersFile /etc/httpd/conf/worker.properties
# Where to put jk logs
JkLogFile /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel debug
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"
# Send all jsp requests to GlassFish
JkMount /*.jsp worker1
# Send all glassfish-test requests to GlassFish
JkMount /glassfish-test/* loadbalancer
```

Example 7-2 `workers.properties` File for Load Balancing

This example shows a `workers.properties` or `glassfish-jk.properties` file that is set for load balancing. The `worker.worker*.port` should match with JK ports you created.

```
worker.list=worker1,worker2,loadbalancer
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=1
worker.worker1.socket_keepalive=1
worker.worker1.socket_timeout=300
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8010
worker.worker2.lbfactor=1
worker.worker2.socket_keepalive=1
worker.worker2.socket_timeout=300
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=worker1,worker2
```

HTTP Load Balancer Deployments

You can configure your load balancer in different ways, depending on your goals and environment, as described in the following sections:

- [“Using Clustered Server Instances” on page 132](#)
- [“Using Multiple Standalone Instances” on page 132](#)

Using Clustered Server Instances

The most common way to deploy the load balancer is with a cluster or clusters of server instances. By default all the instances in a cluster have the same configuration and the same applications deployed to them. The load balancer distributes the workload between the server instances and requests fail over from an unhealthy instance to a healthy one. If you’ve configured HTTP session persistence, session information persists when the request is failed over.

If you have multiple clusters, requests can be load balanced across clusters but are only failed over between the instances in a single cluster. Use multiple clusters in a load balancer to easily enable rolling upgrades of applications. For more information, see [Chapter 8, “Upgrading Applications Without Loss of Availability.”](#)

Note – Requests cannot be load balanced across clusters and standalone instances.

Using Multiple Standalone Instances

It is also possible to configure your load balancer to use multiple standalone instances, and load balance and failover requests between them. However, in this configuration, you must manually ensure that the standalone instances have homogenous environments and the same applications deployed to them. Because clusters automatically maintain a homogenous environment, for most situations it is better and easier to use clusters.

Tip – Load balancing across multiple standalone instances only provides failover for requests, and any associated HTTP session data will not be failed over. This is another reason why using a cluster, which can provide session failover, is a more desirable load balancing configuration.

Upgrading Applications Without Loss of Availability

Upgrading an application to a new version without loss of availability to users is called a *rolling upgrade*. Carefully managing the two versions of the application across the upgrade ensures that current users of the application complete their tasks without interruption, while new users transparently get the new version of the application. With a rolling upgrade, users are unaware that the upgrade occurs.

For more information about application versions and how they are identified, see “Module and Application Versions” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

In a clustered environment, a rolling upgrade redeploys an application with a minimal loss of service and sessions. A session is any artifact that can be replicated, for example:

- HttpSession
- SingleSignOn
- ServletTimer
- DialogFragment
- Stateful session bean

A rolling upgrade can take place under light to moderate loads. The procedure requires about 10-15 minutes for each GlassFish Server instance.



Caution – To prevent the risk of version mismatch when a session fails over, upgrade all instances in a cluster at the same time. Otherwise a session might fail over to an instance where different versions of components are running.

Perform this task on each cluster separately. A cluster acts as a safe boundary for session failover for instances in the cluster. Sessions in one cluster can never fail over to sessions in another cluster. Therefore, the risk of version mismatch is avoided.

Application Compatibility

Rolling upgrades pose varying degrees of difficulty depending on the magnitude of changes between the two application versions.

If the changes are superficial, for example, changes to static text and images, the two versions of the application are *compatible* and can both run at once in the same cluster.

Compatible applications must:

- Use the same session information
- Use compatible database schemas
- Have generally compatible application-level business logic
- Use the same physical data source

You can perform a rolling upgrade of a compatible application in either a single cluster or multiple clusters. For more information, see [“Upgrading In a Single Cluster” on page 134](#).

If the two versions of an application do not meet all the above criteria, then the applications are considered *incompatible*. Executing incompatible versions of an application in one cluster can corrupt application data and cause session failover to not function correctly. The problems depend on the type and extent of the incompatibility. It is good practice to upgrade an incompatible application by creating a “shadow cluster” to which to deploy the new version and slowly quiesce the old cluster and application. For more information, see [“Upgrading Incompatible Applications” on page 136](#).

The application developer and administrator are the best people to determine whether application versions are compatible. If in doubt, assume that the versions are incompatible, since this is the safest approach.

Upgrading In a Single Cluster

You can perform a rolling upgrade of an application deployed to a single cluster, providing the cluster’s configuration is not shared with any other cluster.

▼ To Upgrade an Application in a Single Cluster

- 1 **Deploy the upgraded application to the cluster in a disabled state and with a new version identifier.**

For example:

```
asadmin> asadmin deploy --enabled=false --target myCluster myApp:1.1
```

- 2 Perform the following steps for each server instance in the cluster.
 - a. Quiesce one server instance in the cluster from the load balancer.
Follow these steps:
 - i. Disable the server instance using `asadmin disable-http-lb-server`.
 - ii. Export the load balancer configuration file using `asadmin export-http-lb-config`.
 - iii. Copy the exported configuration file to the web server instance's configuration directory.
For example, for Sun Java System Web Server, the location is `web-server-install-dir/https-host-name/config/loadbalancer.xml`.
 - iv. Wait until the timeout has expired.
Monitor the load balancer's log file.
 - b. Enable the upgraded application version on the quiesced server instance.
For example:

```
asadmin> asadmin enable --target instance01 myApp:1.1
```


Enabling the upgraded application version automatically disables the previous version.
 - c. Test the upgraded application on the server instance to make sure it runs correctly.
 - d. Re-enable the server instance in load balancer.
Follow these steps:
 - i. Enable the server instance using `asadmin enable-http-lb-server`.
 - ii. Export the load balancer configuration file using `asadmin export-http-lb-config`.
 - iii. Copy the configuration file to the web server's configuration directory.

Upgrading in Multiple Clusters

▼ To Upgrade a Compatible Application in Two or More Clusters

Repeat the following procedure for each cluster.

- 1 **Deploy the upgraded application to one cluster in a disabled state and with a new version identifier.**

For example:

```
asadmin> asadmin deploy --enabled=false --target myCluster myApp:1.1
```

- 2 **Quiesce the cluster with the upgraded application from the load balancer.**
 - a. **Disable the cluster using `asadmin disable-http-lb-server`.**
 - b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**
 - c. **Copy the exported configuration file to the web server instance's configuration directory.**

For example, for Sun Java System Web Server, the location is `web-server-install-dir/https-host-name/config/loadbalancer.xml`.
 - d. **Wait until the timeout has expired.**

Monitor the load balancer's log file.

- 3 **Enable the upgraded application version on the quiesced cluster.**

For example:

```
asadmin> asadmin enable --target myCluster myApp:1.1
```

Enabling the upgraded application version automatically disables the previous version.

- 4 **Test the upgraded application on the cluster to make sure it runs correctly.**
- 5 **Enable the cluster in the load balancer:**
 - a. **Enable the cluster using `asadmin enable-http-lb-server`.**
 - b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**
 - c. **Copy the configuration file to the web server's configuration directory.**

Upgrading Incompatible Applications

If the new version of the application is incompatible with the old version, use the following procedure. For information on what makes applications compatible, see [“Application Compatibility” on page 134](#). Also, you must upgrade incompatible application in two or more clusters. If you have only one cluster, create a “shadow cluster” for the upgrade, as described below.

When upgrading an incompatible application:

- Give the new version of the application a different version identifier from the old version of the application. The steps below assume that the application has a new version identifier.
- If the data schemas are incompatible, use different physical data sources after planning for data migration.
- Deploy the new version to a different cluster from the cluster where the old version is deployed.
- Set an appropriately long timeout for the cluster running the old application before you take it offline, because the requests for the application won't fail over to the new cluster. These user sessions will simply fail.

▼ To Upgrade an Incompatible Application by Creating a Second Cluster

- 1 Create a “shadow cluster” on the same or a different set of machines as the existing cluster. If you already have a second cluster, skip this step.
 - a. Use the Administration Console to create the new cluster and reference the existing cluster's named configuration.
Customize the ports for the new instances on each machine to avoid conflict with existing active ports.
 - b. For all resources associated with the cluster, add a resource reference to the newly created cluster using `asadmin create-resource-ref`.
 - c. Create a reference to all other applications deployed to the cluster (except the current upgraded application) from the newly created cluster using `asadmin create-application-ref`.
 - d. Configure the cluster to be highly available using `asadmin configure-ha-cluster`.
 - e. Create reference to the newly-created cluster in the load balancer configuration file using `asadmin create-http-lb-ref`.
- 2 Give the new version of the application a different version identifier from the old version.
- 3 Deploy the new application version with the new cluster as the target. Use a different context root or roots.

- 4 Start the new cluster while the other cluster is still running.**
The start causes the cluster to synchronize with the domain and be updated with the new application.
- 5 Test the application on the new cluster to make sure it runs correctly.**
- 6 Disable the old cluster from the load balancer using `asadmin disable-http-lb-server`.**
- 7 Set a timeout for how long lingering sessions survive.**
- 8 Enable the new cluster from the load balancer using `asadmin enable-http-lb-server`.**
- 9 Export the load balancer configuration file using `asadmin export-http-lb-config`.**
- 10 Copy the exported configuration file to the web server instance's configuration directory.**
For example, for Sun Java System Web Server, the location is
web-server-install-dir/https-host-name/config/loadbalancer.xml.
- 11 After the timeout period expires or after all users of the old application have exited, stop the old cluster and undeploy the old application version.**

Configuring High Availability Session Persistence and Failover

This chapter explains how to enable and configure high availability session persistence.

- [“Overview of Session Persistence and Failover” on page 139](#)
- [“Enabling the High Availability Session Persistence Service” on page 142](#)
- [“Stateful Session Bean Failover” on page 147](#)

Overview of Session Persistence and Failover

GlassFish Server provides high availability session persistence through *failover* of HTTP session data and stateful session bean (SFSB) session data. Failover means that in the event of an server instance or hardware failure, another server instance in a cluster takes over a distributed session.

For example, Java EE applications typically have significant amounts of session state data. A web shopping cart is the classic example of 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.

Note – When using high availability session persistence together with a load balancer, use a load balancer that includes session-based stickiness as part of its load-balancing algorithm. Otherwise, session data can be misdirected or lost. An example of a load balancer that includes session-based stickiness is the Loadbalancer Plug-In available in Oracle GlassFish Server.

The following topics are addressed here:

- [“Requirements” on page 140](#)
- [“Restrictions” on page 140](#)
- [“Scope” on page 141](#)

Requirements

A distributed session can run in multiple Oracle GlassFish Server instances, if:

- Each server instance has access to the same session state data. GlassFish Server supports in-memory session replication on other servers in the cluster for maintaining HTTP session and stateful session bean data. In-memory session replication is enabled by default for GlassFish Server clustered environments if the Group Management Service is enabled.

The use of in-memory replication requires the Group Management Service (GMS) to be enabled. For more information about GMS, see [“Group Management Service” on page 66](#).

If server instances in a cluster are located on different hosts, ensure that the following prerequisites are met:

- To ensure that GMS and in-memory replication function correctly, the hosts must be on the same subnet.
- To ensure that in-memory replication functions correctly, the system clocks on all hosts in the cluster must be synchronized as closely as possible.

Note – GlassFish Server 3.1 does not support High Availability Database (HADB) configurations. Instead, use in-memory replication, as described in [“High Availability Session Persistence” on page 18](#).

- Each server instance has the same distributable web application deployed to it. The web-app element of the web.xml deployment descriptor file must contain the distributable element.
- The web application uses high-availability session persistence. If a non-distributable web application is configured to use high-availability session persistence, the server writes an error to the log file.
- The web application must be deployed using the `deploy` or `deploydir` subcommand with the `--availabilityenabled` option set to `true`. For more information on these subcommands, see `deploy(1)` and `deploydir(1)`.

Restrictions

When configuring session persistence and failover, note the following restrictions:

- When a session fails over, any references to open files or network connections are lost. Applications must be coded with this restriction in mind.
- EJB Singletons are created for each server instance in a cluster, and not once per cluster.

- The high availability session persistence service is not compatible with dynamic deployment, dynamic reloading, and autodeployment. These features are for development, not production environments, so you must disable them before enabling the session persistence service. For information about how to disable these features, see the *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.
- GlassFish Server 3.1 does not support High Availability Database (HADB) configurations. Instead, use in-memory replication, as described in [“High Availability Session Persistence” on page 18](#).
- You can only bind certain objects to distributed sessions that support failover. Contrary to the Servlet 2.4 specification, GlassFish Server 3.1 does not throw an `IllegalArgumentException` if an object type not supported for failover is bound into a distributed session.

You can bind the following objects into a distributed session that supports failover:

- Local home and object references for all EJB components.
- Colocated stateless session, stateful session, or entity bean reference .
- Distributed stateless session, stateful session, or entity bean reference.
- JNDI Context for `InitialContext` and `java:comp/env`.
- `UserTransaction` objects. However, if the instance that fails is never restarted, any prepared global transactions are lost and might not be correctly rolled back or committed.
- Serializable Java types.
- You cannot bind the following object types into sessions that support failover:
 - JDBC `DataSource`
 - Java Message Service (JMS) `ConnectionFactory` and `Destination` objects
 - JavaMail™ `Session`
 - `ConnectionFactory`
 - Administered Objects
 - Web service reference

In general, for these objects, failover will not work. However, failover might work in some cases, if for example the object is serializable.

Scope

The availability service can be enabled for the following scopes, ranging from highest to lowest:

- Cluster
- Standalone server instance (not part of a cluster)
- Web, EJB, or JMS container in a cluster
- Application

- Standalone Web, EJB, or JMS module
- Individual Stateful Session Bean (SFSB)

In general, enabling or disabling availability session persistence for a cluster or container involves setting the boolean `availability-service` property to `true` or `false` by means of the `asadmin set` subcommand. The availability service is enabled by default for GlassFish Server clusters and all Web, EJB, and JMS containers running in a cluster.

The value set for the `availability-service` property is inherited by all child objects running in a given cluster or container unless the value is explicitly overridden at the individual module or application level. For example, if the `availability-service` property is set to `true` for an EJB container, the availability service will be enabled by default for all EJB modules running in that container.

Conversely, to enable availability at a given scope, you must enable it at all higher levels as well. For example, to enable availability at the application level, you must also enable it at the cluster or server instance and container levels.

Enabling the High Availability Session Persistence Service

This section explains how to configure and enable the high availability session persistence service.

- [“To Enable Availability for a Cluster, Standalone Instance or Container” on page 142](#)
- [“Configuring Availability for Individual Web Applications” on page 144](#)
- [“Configuring Replication and Multi-Threaded Concurrent Access to `HttpSessions`” on page 145](#)
- [“Using Single Sign-on with Session Failover” on page 145](#)
- [“Using Coherence*Web for HTTP Session Persistence” on page 147](#)

▼ To Enable Availability for a Cluster, Standalone Instance or Container

This procedure explains how to enable high availability for a cluster as a whole, or for Web, EJB, or JMS containers that run in a cluster, or for a standalone server instance that is not part of a cluster.

1 Create a GlassFish Server cluster.

For more information, see [“To Create a Cluster” on page 75](#).

2 Set up load balancing for the cluster.

For instructions, see [“Setting Up HTTP Load Balancing” on page 129](#).

3 Verify that the cluster and all instances within the cluster for which you want to enable availability is running.

These steps are also necessary when enabling availability for a Web, EJB, or JMS container running in a cluster. The cluster and all instances in the cluster for which you want to enable availability must be running.

a. Verify that the cluster is running.

```
asadmin> list-clusters
```

A list of clusters and their status (running, not running) is displayed. If the cluster for which you want to enable availability is not running, you can start it with the following command:

```
asadmin> start-cluster cluster-name
```

b. Verify that all instances in the cluster are running.

```
asadmin> list-instances
```

A list of instances and their status is displayed. If the instances for which you want to enable availability are not running, you can start them by using the following command for each instance:

```
asadmin> start-instance instance-name
```

4 Use one of the following asadmin set(1) subcommands to enable availability for a specific cluster, or for a specific Web, EJB, or JMS container.

■ For a cluster as a whole

```
asadmin> set cluster-name-config.availability-service.availability-enabled=true
```

For example, for a cluster named c1:

```
asadmin> set c1-config.availability-service.availability-enabled=true
```

■ For the Web container in a cluster

```
asadmin> set cluster-name-config.availability-service \
.web-container-availability.availability-enabled=true
```

■ For the EJB container in a cluster

```
asadmin> set cluster-name-config.availability-service \
.ejb-container-availability.availability-enabled=true
```

■ For the JMS container in a cluster

```
asadmin> set cluster-name-config.availability-service \
.jms-availability.availability-enabled=true
```

■ For a standalone server instance (not part of a cluster)

```
asadmin> set instance-name-config.availability-service.availability-enabled=true
```

5 Restart the standalone server instance or each server instance in the cluster.**6 Enable availability for any SFSB that requires it.**

Select methods for which checkpointing the session state is necessary. For more information, see [“Configuring Availability for an Individual Bean” on page 149](#).

7 Make each web module distributable if you want it to be highly available.

For more information, see “Web Module Deployment Guidelines” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

8 (Optional) Enable availability for individual applications, web modules, or EJB modules during deployment.

See the links below for instructions.

- See Also**
- [“Configuring Availability for Individual Web Applications” on page 144](#)
 - [“Using Single Sign-on with Session Failover” on page 145](#)

Configuring Availability for Individual Web Applications

To enable and configure availability for an individual web application, edit the application deployment descriptor file, `glassfish-web.xml`. The settings in an application’s deployment descriptor override the web container’s availability settings.

The `session-manager` element’s `persistence-type` attribute determines the type of session persistence an application uses. It must be set to `replicated` to enable high availability session persistence.

Example

```
<glassfish-web-app> ...
  <session-config>
    <session-manager persistence-type="replicated">
      <manager-properties>
        <property name="persistenceFrequency" value="web-method" />
      </manager-properties>
      <store-properties>
        <property name="persistenceScope" value="session" />
      </store-properties>
    </session-manager> ...
  </session-config> ...
```


Configuring Replication and Multi-Threaded Concurrent Access to HttpSession

If you are using Memory Replication and your web application involves multiple client threads concurrently accessing the same session ID, then you may experience session loss even without any instance failure. The problem is that the GlassFish Server 3.1 memory replication framework makes use of session versioning. This feature was designed with the more traditional HTTP request/response communication model in mind.

However, for some types of applications, the traditional request/response model does not work. Examples include many Ajax-related frameworks and the use of Frames. Another example is when a page includes many static resources, such as JPG files. In these situations, most browsers will optimize the loading of these resources by using multiple parallel connections, each of which is handled by a separate request processing thread. If the application has already established a session, then this will also involve more than one thread at a time accessing a single HttpSession.

The solution in such cases is to use the `relaxVersionSemantics` property in the `glassfish-web.xml` deployment descriptor file for the application. This enables the web container to return for each requesting thread whatever version of the session that is in the active cache regardless of the version number. This is critical when multiple threads are interacting in an essentially non-deterministic fashion with the container.

Example

The following is an example snippet from a `glassfish-web.xml` file that illustrates where to add the `relaxVersionSemantics` property.

```
<glassfish-web-app>
  <session-config>
    <session-manager persistence-type="replicated">
      <manager-properties>
        <property name="relaxCacheVersionSemantics" value="true"/>
      </manager-properties>
    </session-manager>
  </session-config>
  . . . . .
</glassfish-web-app>
```

Using Single Sign-on with Session Failover

In a single application server instance, once a user is authenticated by an application, the user is not required to re-authenticate individually to other applications running on the same instance. This is called *single sign-on*.

For this feature to continue to work even when an HTTP session fails over to another instance in a cluster, single sign-on information must be persisted using in-memory replication. To persist single sign-on information, first, enable availability for the server instance and the web container, then enable single-sign-on state failover.

You can enable single sign-on state failover by using the `asadmin set` command to set the configuration's `availability-service.web-container-availability.sso-failover-enabled` property to `true`.

For example, use the `set` command as follows, where `config1` is the configuration name:

```
asadmin> set config1.availability-service.web-container-availability. \
sso-failover-enabled="true"
```

Single Sign-On Groups

Applications that can be accessed through a single name and password combination constitute a *single sign-on group*. For HTTP sessions corresponding to applications that are part of a single sign-on group, if one of the sessions times out, other sessions are not invalidated and continue to be available. This is because time out of one session should not affect the availability of other sessions.

As a corollary of this behavior, if a session times out and you try to access the corresponding application from the same browser window that was running the session, you are not required to authenticate again. However, a new session is created.

Take the example of a shopping cart application that is a part of a single sign-on group with two other applications. Assume that the session time out value for the other two applications is higher than the session time out value for the shopping cart application. If your session for the shopping cart application times out and you try to run the shopping cart application from the same browser window that was running the session, you are not required to authenticate again. However, the previous shopping cart is lost, and you have to create a new shopping cart. The other two applications continue to run as usual even though the session running the shopping cart application has timed out.

Similarly, suppose a session corresponding to any of the other two applications times out. You are not required to authenticate again while connecting to the application from the same browser window in which you were running the session.

Note – This behavior applies only to cases where the session times out. If single sign-on is enabled and you invalidate one of the sessions using `HttpSession.invalidate()`, the sessions for all applications belonging to the single sign-on group are invalidated. If you try to access any application belonging to the single sign-on group, you are required to authenticate again, and a new session is created for the client accessing the application.

Using Coherence*Web for HTTP Session Persistence

Built on top of Oracle Coherence, Coherence*Web is an HTTP session management module dedicated to managing session state in clustered environments. Starting with Coherence 3.7 and GlassFish Server 3.1, there is a new feature of Coherence*Web called ActiveCache for GlassFish. ActiveCache for GlassFish provides Coherence*Web functionality in web applications deployed on GlassFish Servers. Within GlassFish Server, Coherence*Web functions as an additional web container persistence type, named coherence-web.

For information about how to configure and deploy Coherence*Web on GlassFish Server, see [Using Coherence*Web with GlassFish Server \(http://download.oracle.com/docs/cd/E18686_01/coh.37/e18690/glassfish.htm\)](http://download.oracle.com/docs/cd/E18686_01/coh.37/e18690/glassfish.htm).

Stateful Session Bean Failover

Stateful session beans (SFSBs) contain client-specific state. There is a one-to-one relationship between clients and the stateful session beans. At creation, the EJB container gives each SFSB a unique session ID that binds it to a client.

An SFSB's state can be saved in a persistent store in case a server instance fails. The state of an SFSB is saved to the persistent store at predefined points in its life cycle. This is called

checkpointing. If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back.

However, if an SFSB participates in a bean-managed transaction, the transaction might be committed in the middle of the execution of a bean method. Since the bean's state might be undergoing transition as a result of the method invocation, this is not an appropriate time to checkpoint the bean's state. In this case, the EJB container checkpoints the bean's state at the end of the corresponding method, provided the bean is not in the scope of another transaction when that method ends. If a bean-managed transaction spans across multiple methods, checkpointing is delayed until there is no active transaction at the end of a subsequent method.

The state of an SFSB is not necessarily transactional and might be significantly modified as a result of non-transactional business methods. If this is the case for an SFSB, you can specify a list of checkpointed methods, as described in [“Specifying Methods to Be Checkpointed” on page 149](#)

If a distributable web application references an SFSB, and the web application's session fails over, the EJB reference is also failed over.

If an SFSB that uses session persistence is undeployed while the GlassFish Server instance is stopped, the session data in the persistence store might not be cleared. To prevent this, undeploy the SFSB while the GlassFish Server instance is running.

Configuring Availability for the EJB Container

To enable availability for the EJB container use the `asadmin set` command to set the following three properties for the configuration:

- `availability-service.ejb-container-availability.availability-enabled`
- `availability-service.ejb-container-availability.sfsb-persistence-type`
- `availability-service.ejb-container-availability.sfsb-ha-persistence-type`

For example, if `config1` is the configuration name, use the following commands:

```
asadmin> set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.availability-enabled="true"

asadmin> set --user admin --passwordfile password.txt --host localhost --port
4849
config1.availability-service.
ejb-container-availability.sfsb-persistence-type="file"
asadmin> set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.sfsb-ha-persistence-type="replicated"
```

Configuring the SFSB Session Store When Availability Is Disabled

If availability is disabled, the local file system is used for SFSB state passivation, but not persistence. To change where the SFSB state is stored, change the Session Store Location setting in the EJB container. For information about configuring store properties, see the Administration Console online help.

Configuring Availability for an Individual Application or EJB Module

You can enable SFSB availability for an individual application or EJB module during deployment:

- If you are deploying with the Administration Console, check the Availability Enabled checkbox.
- If you are deploying using the `asadmin deploy` or `asadmin deploydir` commands, set the `--availabilityenabled` option to `true`. For more information, see `deploy(1)` and `deploydir(1)`.

Configuring Availability for an Individual Bean

To enable availability and select methods to be checkpointed for an individual SFSB, use the `glassfish-ejb-jar.xml` deployment descriptor file.

To enable high availability session persistence, set `availability-enabled="true"` in the `ejb` element.

EXAMPLE 9-1 Example of an EJB Deployment Descriptor With Availability Enabled

```
<glassfish-ejb-jar>
...
  <enterprise-beans>
    ...
    <ejb availability-enabled="true">
      <ejb-name>MySFSB</ejb-name>
    </ejb>
    ...
  </enterprise-beans>
</glassfish-ejb-jar>
```

Specifying Methods to Be Checkpointed

If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back. To specify additional optional checkpointing of SFSBs at the end of non-transactional business methods that cause important modifications to the bean's state, use the `checkpoint-at-end-of-method` element in the `ejb` element of the `glassfish-ejb-jar.xml` deployment descriptor file.

The non-transactional methods in the `checkpoint-at-end-of-method` element can be:

- `create()` methods defined in the home interface of the SFSB, if you want to checkpoint the initial state of the SFSB immediately after creation
- For SFSBs using container managed transactions only, methods in the remote interface of the bean marked with the transaction attribute `TX_NOT_SUPPORTED` or `TX_NEVER`
- For SFSBs using bean managed transactions only, methods in which a bean managed transaction is neither started nor committed

Any other methods mentioned in this list are ignored. At the end of invocation of each of these methods, the EJB container saves the state of the SFSB to persistent store.

Note – If an SFSB does not participate in any transaction, and if none of its methods are explicitly specified in the `checkpoint-at-end-of-method` element, the bean's state is not checkpointed at all even if `availability-enabled="true"` for this bean.

For better performance, specify a *small* subset of methods. The methods should accomplish a significant amount of work or result in important modification to the bean's state.

EXAMPLE 9-2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing

```
<glassfish-ejb-jar>
...
  <enterprise-beans>
    ...
    <ejb availability-enabled="true">
      <ejb-name>ShoppingCartEJB</ejb-name>
      <checkpoint-at-end-of-method>
        <method>
          <method-name>addToCart</method-name>
        </method>
      </checkpoint-at-end-of-method>
    </ejb>
    ...
  </enterprise-beans>
</glassfish-ejb-jar>
```

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” on page 151](#)
- [“Connection Failover” on page 158](#)
- [“Load-Balanced Delivery to MDBs” on page 160](#)

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” on page 152](#)
- [“Configuring GlassFish Clusters to Use Message Queue Broker Clusters” on page 153](#)
- [“To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster With Master Broker” on page 154](#)
- [“To Configure a GlassFish Cluster to Use an Embedded or Local Conventional Broker Cluster of Peer Brokers” on page 155](#)
- [“To Change the Master Broker in an Embedded or Local Broker Cluster” on page 156](#)
- [“To Migrate Between Types of Embedded or Local Conventional Broker Clusters” on page 156](#)
- [“To Configure a GlassFish Cluster to Use a Local Enhanced Broker Cluster” on page 156](#)
- [“To Configure a GlassFish Cluster to Use a Remote Broker Cluster” on page 157](#)

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 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 *Open 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 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 156. 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 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 156. 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 Although not an absolute requirement, you should make sure all GlassFish instances and Message Queue brokers in the cluster are running before using the `change-master-broker` command in order to avoid later internal configuration synchronization of any unavailable instance or 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.



Caution – Before using this procedure to reconfigure an existing cluster, you must follow the special procedures to migrate from a conventional broker cluster to an enhanced broker cluster, as described in “Converting a Conventional Cluster to an Enhanced Cluster” in *Open Message Queue 4.5 Administration Guide*. 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 *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 82 and “To Create an Instance Locally” on page 92.**

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 *Open 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 *GlassFish Server Open Source Edition 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 *Open Message Queue 4.5 Administration Guide*.

RMI-IIOP Load Balancing and Failover

This chapter describes using high-availability features for remote EJB references and JNDI objects over RMI-IIOP in GlassFish Server.

- [“Overview” on page 161](#)
- [“InitialContext Load Balancing” on page 162](#)
- [“Per-Request Load Balancing \(PRLB\)” on page 166](#)

Overview

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

The following topics are addressed here:

- [“General Requirements for Configuring Load Balancing” on page 161](#)
- [“Load Balancing Models” on page 162](#)

General Requirements for Configuring Load Balancing

Oracle GlassFish Server provides high availability of remote EJB references and `NameService` objects over RMI-IIOP, provided all the following apply:

- Your deployment has a cluster of at least two instances.
- Java EE applications are deployed to all instances and clusters that participate in load balancing.
- RMI-IIOP client applications are enabled for load balancing.

GlassFish Server supports load balancing for Java applications executing in the Application Client Container (ACC). See [“Enabling RMI-IIOP Hardware Load Balancing and Failover” on page 163](#).

Note – GlassFish Server does not support RMI-IIOP load balancing and failover over secure sockets layer (SSL).

Load Balancing Models

GlassFish Server supports two general models for load balancing:

[“InitialContext Load Balancing” on page 162](#)

When a client performs a JNDI lookup for an object, the Naming Service creates a `InitialContext` (IC) object associated with a particular server instance. From then on, all lookup requests made using that IC object are sent to the same server instance. `InitialContext` load balancing can be configured automatically across an entire cluster.

[“Per-Request Load Balancing \(PRLB\)” on page 166](#)

Per Request Load Balancing (PRLB) is a method for load balancing stateless EJBs that enables load-balancing for each request to an EJB instance. PRLB chooses the first node in a cluster to use on each request. PRLB is configured on a per-EJB basis.

InitialContext Load Balancing

The following topics are addressed here:

- [“InitialContext Summary” on page 162](#)
- [“InitialContext Algorithm” on page 163](#)
- [“Enabling RMI-IIOP Hardware Load Balancing and Failover” on page 163](#)

InitialContext Summary

When `InitialContext` load balancing is used, the client calls the `InitialContext()` method to create a new `InitialContext` (IC) object that is associated with a particular server instance. JNDI lookups are then performed on that IC object, and all lookup requests made using that IC object are sent to the same server instance. All `EJBHome` objects looked up with that `InitialContext` are 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 live target servers when creating `InitialContext` objects. If the target server instance goes down, the lookup or EJB method invocation will failover to another server instance. All objects derived from same `InitialContext` will failover to the same server instance.

IIOP load balancing and failover happens transparently. No special steps are needed during application deployment. IIOP load balancing and failover for the GlassFish Server supports dynamically reconfigured clusters. 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. Therefore, you are not required to manually update the list of endpoints if a new instance is added to the cluster or deleted from the cluster. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

InitialContext Algorithm

GlassFish Server uses a randomization and round-robin algorithm for RMI-IIOP load balancing and failover.

When an RMI-IIOP client first creates a new `InitialContext` object, the list of available GlassFish Server IIOP endpoints is randomized for that client. For that `InitialContext` object, the load balancer directs lookup requests and other `InitialContext` operations to an endpoint on the randomized list. If that endpoint is not available then a different random endpoint in the list is used.

Each time the client subsequently creates a new `InitialContext` object, the endpoint list is rotated so that a different IIOP endpoint is used for `InitialContext` operations. The rotation is randomized, so the rotation is not to the next endpoint in the list, but instead to a random endpoint in the list.

When you obtain or create beans from references obtained by an `InitialContext` object, those beans are created on the GlassFish Server instance serving the IIOP endpoint assigned to the `InitialContext` object. The references to those beans contain the IIOP endpoint addresses of all GlassFish Server instances in the cluster.

The *primary endpoint* is the bean endpoint corresponding to the `InitialContext` endpoint used to look up or create the bean. The other IIOP endpoints in the cluster are designated as *alternate endpoints*. If the bean's primary endpoint becomes unavailable, further requests on that bean fail over to one of the alternate endpoints.

You can configure RMI-IIOP load balancing and failover to work with applications running in the ACC.

Enabling RMI-IIOP Hardware Load Balancing and Failover

You can enable RMI-IIOP load balancing and failover for applications running in the application client container (ACC). Weighted round-robin load balancing is also supported.

▼ To Enable RMI-IIOP Hardware Load Balancing for the Application Client Container

This procedure provides an overview of the steps necessary to enable RMI-IIOP load balancing and failover with the application client container (ACC). For additional information on the ACC, see “Developing Clients Using the ACC” in *GlassFish Server Open Source Edition 3.1 Application Development Guide*.

Before You Begin The first five steps in this procedure are only necessary if you are enabling RMI-IIOP load balancing on a system other than the DAS. This is common in production environment, but less common in a development environment. For example, a developer who wants to experiment with a cluster and load balancing might create two instances on the same system on which the DAS is running. In such cases, steps 1-5 are unnecessary.

1 Go to the `install_dir/bin` directory.

2 Run `package-appclient`.

This utility produces an `appclient.jar` file. For more information on `package-appclient`, see `package-appclient(1M)`.

3 Copy the `appclient.jar` file to the machine where you want your client and extract it.

4 Edit the `asenv.conf` or `asenv.bat` path variables to refer to the correct directory values on that machine.

The file is at `appclient-install-dir/config/`.

For a list of the path variables to update, see `package-appclient(1M)`.

5 If required, make the `appclient` script executable.

For example, on UNIX use `chmod 700`.

6 Find the IIOP listener port number for at least two instances in the cluster.

You specify the IIOP listeners as endpoints in [Step 7](#).

For each instance, obtain the IIOP listener ports as follows:

a. Verify that the instances for which you want to determine the IIOP listener port numbers are running.

```
asadmin> list-instances
```

A list of instances and their status (running, not running) is displayed.

The instances for which you want to display the IIOP listener ports must be running.

- b. For each instance, enter the following command to list the various port numbers used by the instance.

```
asadmin> get servers.server.instance-name.system-property.*.value
```

For example, for an instance name `in1`, you would enter the following command:

```
asadmin> get servers.server.in1.system-property.*.value
```

7 Add at least two target-server elements in the `sun-acc.xml` file.

Use the endpoints that you obtained in [Step 6](#).

If the GlassFish Server instance on which the application client is deployed participates in a cluster, the ACC 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.

The `target-server` element specifies one or more IIOP endpoints used for load balancing. The `address` attribute is an IPv4 address or host name, and the `port` attribute specifies the port number. See “client-container” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

As an alternative to using `target-server` elements, you can use the `endpoints` property as follows:

```
jvmarg value = "-Dcom.sun.appserv.iiop.endpoints=host1:port1,host2:port2,..."
```

8 If you require weighted round-robin load balancing, perform the following steps:

- a. Set the load-balancing weight of each server instance.

```
asadmin set instance-name.lb-weight=weight
```

- b. In the `sun-acc.xml`, set the `com.sun.appserv.iiop.loadbalancingpolicy` property of the ACC to `ic-based-weighted`.

```
...
<client-container send-password="true">
  <property name="com.sun.appserv.iiop.loadbalancingpolicy" \
    value="ic-based-weighted"/>
...
```

9 Deploy your client application with the `--retrieve` option to get the client jar file.

Keep the client jar file on the client machine.

For example:

```
asadmin --user admin --passwordfile pw.txt deploy --target cluster1 \
--retrieve my_dir myapp.ear
```

10 Run the application client as follows:

```
appclient --client my_dir/myapp.jar
```

Example 11–1 Setting Load-Balancing Weights for RMI-IIOP Weighted Round-Robin Load Balancing

In this example, the load-balancing weights in a cluster of three instances are to be set as shown in the following table.

Instance Name	Load-Balancing Weight
i1	100
i2	200
i3	300

The sequence of commands to set these load balancing weights is as follows:

```
asadmin set i1.lb-weight=100
asadmin set i2.lb-weight=200
asadmin set i3.lb-weight=300
```

Next Steps To test failover, stop one instance in the cluster and see that the application functions normally. You can also have breakpoints (or sleeps) in your client application.

To test load balancing, use multiple clients and see how the load gets distributed among all endpoints.

See Also See [“Enabling the High Availability Session Persistence Service” on page 142](#) for instructions on enabling the session availability service for a cluster or for a Web, EJB, or JMS container running in a cluster.

Per-Request Load Balancing (PRLB)

The following topics are addressed here:

- [“PRLB Summary” on page 166](#)
- [“Enabling Per-Request Load Balancing” on page 167](#)

PRLB Summary

Per Request Load Balancing (PRLB) is a method for load balancing stateless EJBs that enables load-balancing for each request to an EJB instance. PRLB chooses the first node in a cluster to use on each request. By contrast, `InitialContext` (hardware) load balancing chooses the first node to use when the `InitialContext` is created, and each request thereafter uses the same node unless a failure occurred.

PRLB is enabled by means of the boolean `per-request-load-balancing` property in the `glassfish-ejb-jar.xml` deployment descriptor file for the EJB. If this property is not set, the original load balancing behavior is preserved.

Note – PRLB is only supported for stateless session beans. Using PRLB with any other bean types will result in a deployment error.

Enabling Per-Request Load Balancing

You can enable Per-Request Load Balancing (PRLB) by setting the boolean `per-request-load-balancing` property to `true` in the `glassfish-ejb-jar.xml` deployment descriptor file for the EJB. On the client side, the `initContext.lookup` method is used to access the stateless EJB.

▼ To Enable RMI-IIOP Per-Request Load Balancing for a Stateless EJB

This procedure describes how to enable PRLB for a stateless EJB that is deployed to clustered GlassFish Server instances. This procedure also provides an client-side example for accessing a stateless EJB that uses PRLB.

1 Choose or assemble the EJB that you want to deploy.

In this example, an EJB named `TheGreeter` is used.

For instructions on developing and assembling an EJB for deployment to GlassFish Server, refer to the following documentation:

- Chapter 8, “Using Enterprise JavaBeans Technology,” in *GlassFish Server Open Source Edition 3.1 Application Development Guide*
- “EJB Module Deployment Guidelines” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*
- “Assembling and Deploying an Application Client Module” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*

2 Set the `per-request-load-balancing` property to `true` in the `glassfish-ejb-jar.xml` deployment descriptor file for the EJB.

For more information about the `glassfish-ejb-jar.xml` deployment descriptor file, refer to “The `glassfish-ejb-jar.xml` File” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*

For example, the `glassfish-ejb-jar.xml` file for a sample EJB named `TheGreeter` is listed below.

```
<glassfish-ejb-jar>
  <enterprise-beans>
    <unique-id>1</unique-id>
```

```
<ejb>
  <ejb-name>TheGreeter</ejb-name>
  <jndi-name>greeter</jndi-name>
  <per-request-load-balancing>true</per-request-load-balancing>
</ejb>
</enterprise-beans>
</glassfish-ejb-jar>
```

3 Deploy the EJB.

If the EJB was previously deployed, it must be redeployed.

For instructions on deploying EJBs, refer to the following documentation:

- “To Deploy an Application or Module” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*
- “To Redeploy an Application or Module” in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*

4 (Optional) Verify the PRLB configuration by looking for the following FINE message in the CORBA log file:

Setting per-request-load-balancing policyfor EJB *EJB-name*

5 Configure a client application to access the PRLB-enabled EJB.

For example:

```
public class EJBClient {
    public static void main(String args[]) {
        :
        :
        :
        try {
            // only one lookup

            Object objref = initContext.lookup("test.cluster.loadbalancing.ejb.\
            TestSessionBeanRemote");
            myGreeterRemote = (TestSessionBeanRemote)PortableRemoteObject.narrow\
            (objref,
                                                    TestSessionBeanRemote.class);

        } catch (Exception e) {
            :
        }

        for (int i=0; i < 10; i++ ) {
            // method calls in a loop.
            String theMessage = myGreeterRemote.sayHello(Integer.toString(i));
            System.out.println("got+": " + theMessage);
        }
    }
}
```


See Also See [“Enabling the High Availability Session Persistence Service” on page 142](#) for instructions on enabling the session availability service for a cluster or for a Web, EJB, or JMS container running in a cluster.

Index

A

- ActiveCache for GlassFish, 147
- algorithm, RMI-IIOP failover, 163
- aliases
 - password
 - encrypted key file authentication, 43
 - password authentication, 45
- alternate endpoints, RMI-IIOP failover, 163
- applibs directory, default resynchronization, 103
- application files, preventing deletion during resynchronization, 111
- applications, upgrading without loss of availability, 133
- applications directory, default resynchronization, 101
- authentication
 - SSH
 - encrypted public key, 42–45
 - overview, 39
 - password, 45–46
 - unencrypted public key, 40–42
- automatic migration, EJB timers, 114–115
- automatically created configurations, 118
- availability, 139–150
 - configuration requirements, 161
 - EJB container level, 149
 - enabling, 142–147
 - for stateful session beans, 147
 - for web modules, 140
 - overview, 139–142
 - requirements, 140
 - restrictions, 140–141

availability (*Continued*)

- scope, 141–142

B

- .bashrc file, 53

C

- central repository
 - instance resynchronization with
 - default behavior, 101–104
 - offline, 111–114
 - online, 104–106
- checkpoint-at-end-of-method element, 149
- checkpointing, 147
 - selecting methods for, 147, 149
- classes directory, default resynchronization, 103
- clusters
 - adding instances to
 - centrally, 82–85
 - locally, 92–96
 - configuring GMS for, 75–77
 - creating, 75–77
 - deleting, 78–80
 - introduction to, 65–66
 - listing, 78
 - listing for configurations, 121–122
 - removing instances from
 - centrally, 86–87
 - locally, 96–98

clusters (*Continued*)

- restarting centrally
 - individual instances in, 91–92
- restarting locally
 - individual instances in, 100–101
- starting centrally
 - all instances in, 87–88
 - individual instances in, 89–90
- starting locally
 - individual instances in, 98–99
- stopping centrally
 - all instances in, 88–89
 - individual instances in, 90–91
- stopping locally
 - individual instances in, 99–100
- cmd.exe command shell, 34
- Coherence*Web, 147
- config directory, default resynchronization, 102
- config-files file, 111
- CONFIG nodes
 - changing type, 62–63
 - creating, 56–58
 - definition, 51
 - deleting, 59–60
 - listing, 58–59
 - updating, 61
- configuration files, adding to list for
 - resynchronization, 110–111
- configurations, *See* named configurations
- configuring
 - SSH server daemon
 - Cygwin, 30–31
 - Linux operating system, 37
 - MacOS operating system, 37
 - MKS Toolkit, 34–35
 - Oracle Solaris operating system, 36
- conventional broker clusters, defined, 152
- copying, GlassFish Server software to multiple
 - hosts, 47–48
- create-password-alias subcommand
 - encrypted key file authentication, 43
 - password authentication, 45
- creating
 - clusters, 75–77

creating (*Continued*)

- configurations, 119–120
 - instances
 - centrally, 82–85
 - locally, 92–96
 - nodes
 - CONFIG, 56–58
 - SSH, 52–53
 - .cshrc file, 53
 - custom configuration files,
 - resynchronization, 108–109
- Cygwin
- downloading, 28–29
 - home directories, 30
 - installing, 28–29
 - OpenSSH, 29
 - paths, 29
 - PubkeyAuthentication option, 31
 - setup.exe file, 28
 - SSH server daemon, 30–31
 - StrictModes option, 31

D

- DAS (domain administration server)
- resynchronization with instances
 - default behavior, 101–104
 - offline, 111–114
 - online, 104–106
- default-config configuration, 118
- default resynchronization, instances and the
- DAS, 101–104
- deleting
- clusters, 78–80
 - configurations, 122–123
 - instances
 - centrally, 86–87
 - locally, 96–98
 - nodes
 - CONFIG, 59–60
 - SSH, 55–56
- deployment, rolling upgrade, 133
- directories
- configuration synchronization, 118–119

directories (*Continued*)

home

Cygwin, 30

MKS Toolkit, 32–34

distributed HTTP sessions, 140

docroot directory, default resynchronization, 102

domain administration server (DAS)

resynchronization with instances

default behavior, 101–104

offline, 111–114

online, 104–106

domain.xml file, default resynchronization, 102

E

EJB timers, migrating, 114–115

encrypted public key authentication, 42–45

endpoints, RMI-IIOP failover, 163

enhanced broker clusters, defined, 152

ext directory, default resynchronization, 103

F

failover

for web module sessions, 140

of stateful session bean state, 147

RMI-IIOP requirements, 161

failure detection, GMS, 66–68

G

generated directory, default resynchronization, 103

GlassFish Server software

installing on multiple hosts, 47–48

uninstalling from multiple hosts, 48–49

GMS, 66–74

configuring for clusters, 75–77

settings, 66–68

group-discovery-timeout-in-millis setting, 67

Group Management Service, *See* GMS**H**

HADB, 140

heartbeat-frequency-in-millis setting, 67

high availability, 17–24

HADB, 140

JMS, 18–19

mod_jk, 17–18

overview, 17–19

RMI-IIOP, 19

session persistence, 18

home directories

Cygwin, 30

MKS Toolkit, 32–34

HOME environment variable, 34

HOMEDRIVE environment variable, 33

HOMEPATH environment variable, 33

HTTP sessions, 18

distributed, 140

I

Illegal sftp packet len message, 53

install-node

subcommand, 47, 48

installing, GlassFish Server software on multiple

hosts, 47–48

instances

creating

centrally, 82–85

locally, 92–96

deleting

centrally, 86–87

locally, 96–98

effect of configurations on, 126–128

listing, 85–86

listing for configurations, 121–122

ports, 126–128

restarting individually

centrally, 91–92

locally, 100–101

resynchronization with the DAS

default behavior, 101–104

offline, 111–114

online, 104–106

instances (*Continued*)

- starting in clusters
 - centrally, 87–88
- starting individually
 - centrally, 89–90
 - locally, 98–99
- stopping in clusters
 - centrally, 88–89
- stopping individually
 - centrally, 90–91
 - locally, 99–100
- system properties, 126–128

J

- java-web-start directory, default resynchronization, 103

JDK software

- paths
 - Cygwin, 29
 - MKS Toolkit, 32

JMS, 18–19**K**

- key file authentication
 - encrypted, 42–45
 - unencrypted, 40–42

L

- lib directory, default resynchronization, 103
- library files, resynchronization, 106–108

Linux operating system

- obtaining SSH software for, 27
- PubkeyAuthentication option, 37
- SSH server daemon, 37
- StrictModes option, 37

listing

- clusters, 78
- configuration targets, 121–122
- configurations, 120–121

listing (*Continued*)

- instances, 85–86
- nodes
 - CONFIG, 58–59
 - SSH, 53–54
- load balancing
 - mod_jk, 130–131
 - per-request (PRLB), 166–169
 - RMI-IIOP requirements, 161
- .login file, 53

M

- MacOS operating system, SSH server daemon, 37

- max-missed-heartbeats setting, 67

- migrating, EJB timers, 114–115

MKS Toolkit

- cmd.exe command shell, 34
- Cygwin, 32
- home directories, 32–34
- installing, 32
- OpenSSH, 32
- paths, 32
- SSH server daemon, 34–35
- userinfo command, 33, 34
- mod_jk, 130–131
 - load balancing with, 17–18

N**named configurations**

- about, 117–119
- automatic creation, 118
- creating, 119–120
- default-config, 118
- deleting, 122–123
- effect on instance values, 126–128
- listing, 120–121
- listing targets of, 121–122
- modifying properties of, 125–126
- ports, 124–125
- properties, 124–125
- shared, 118

named configurations (*Continued*)

- standalone, 117
- synchronization, 102, 118–119

names, automatically created configurations, 118

nodes

- CONFIG
 - changing type, 62–63
 - creating, 56–58
 - definition, 51
 - deleting, 59–60
 - listing, 58–59
 - updating, 61
- SSH
 - changing type, 62–63
 - creating, 52–53
 - definition, 51
 - deleting, 55–56
 - listing, 53–54
 - testing, 54–55
 - updating, 60–61

types, 51–52

O

offline resynchronization, instances and the DAS, 111–114

online resynchronization, instances and the DAS, 104–106

OpenSSH, Cygwin, 29

Oracle Solaris operating system

- PubkeyAuthentication option, 36
- SSH server daemon, 36
- StrictModes option, 36

P

passphrases, key files, 42

password aliases

- encrypted key file authentication, 43
- password authentication, 45

password authentication, 45–46

passwords, SSH, 45–46

paths

- Cygwin, 29
- JDK software
 - Cygwin, 29
 - MKS Toolkit, 32
 - MKS Toolkit, 32

per-request load balancing (PRLB), 166–169

persistence, 139–150

- Coherence*Web, 147
- enabling, 142–147
- overview, 139–142
- requirements, 140
- restrictions, 140–141
- scope, 141–142

persistence, session, 18

persistence store, for stateful session bean state, 147

ports

- configurations, 124–125
- instances, 126–128

primary endpoints, RMI-IIOP failover, 163

.profile file, 53

properties, configurations, 124–125

PubkeyAuthentication option

- Cygwin, 31
- Linux operating system, 37
- Oracle Solaris operating system, 36

public key authentication

- password, 45–46
- unencrypted, 40–42, 42–45

R

redployment, rolling upgrade, 133

removing, GlassFish Server software from multiple hosts, 48–49

restarting

- instances individually
 - centrally, 91–92
 - locally, 100–101

resynchronization

- additional configuration files, 110–111
- custom configuration files, 108–109
- instances and the DAS
 - default behavior, 101–104

- resynchronization, instances and the DAS (*Continued*)
 - offline, 111–114
 - online, 104–106
 - library files, 106–108
 - preventing for application files, 111
- RMI-IIOP, 19
- rolling upgrade, 133

S

- secure shell (SSH)
 - advantages for centralized administration, 26–27
 - authentication
 - encrypted public key, 42–45
 - overview, 39
 - password, 45–46
 - unencrypted public key, 40–42
 - Cygwin setup, 28–31
 - disadvantages for centralized administration, 26–27
 - MKS Toolkit setup, 31–35
 - obtaining software, 27
 - overview, 26–28
 - server daemon
 - Cygwin, 30–31
 - Linux operating system, 37
 - MacOS operating system, 37
 - MKS Toolkit, 34–35
 - Oracle Solaris operating system, 36
 - testing setup, 38–39
 - user for, 27–28
- server daemon
 - Cygwin, 30–31
 - Linux operating system, 37
 - MacOS operating system, 37
 - MKS Toolkit, 34–35
 - Oracle Solaris operating system, 36
- servers, relation to clusters, 65–66
- session persistence, 139–150
 - and single sign-on, 145–147
 - Coherence*Web, 147
 - enabling, 142–147
 - for stateful session beans, 147, 148
 - for web modules, 140
 - overview, 139–142

- session persistence (*Continued*)
 - requirements, 140
 - restrictions, 140–141
 - scope, 141–142
- session store, for stateful session beans, 148
- sessions
 - HTTP, 18
 - persistence, 18
- settings, GMS failure detection, 66–68
- setup.exe file, Cygwin, 28
- setup-ssh subcommand, 40, 42
- shared configurations, 118
- single sign-on, and session persistence, 145–147
- software
 - installing on multiple hosts, 47–48
 - uninstalling from multiple hosts, 48–49
- SSH (secure shell)
 - authentication
 - encrypted public key, 42–45
 - overview, 39
 - password, 45–46
 - unencrypted public key, 40–42
 - Cygwin setup, 28–31
 - MKS Toolkit setup, 31–35
 - obtaining software, 27
 - testing setup, 38–39
 - user for, 27–28
- SSH (secure shell)
 - advantages for centralized administration, 26–27
 - disadvantages for centralized administration, 26–27
 - overview, 26–28
 - server daemon
 - Cygwin, 30–31
 - Linux operating system, 37
 - MacOS operating system, 37
 - MKS Toolkit, 34–35
 - Oracle Solaris operating system, 36
- ssh-keygen utility, 42
- SSH nodes
 - changing type, 62–63
 - creating, 52–53
 - definition, 51
 - deleting, 55–56
 - listing, 53–54

SSH nodes (*Continued*)

- testing, 54–55
- updating, 60–61

sshd daemon

- Cygwin, 30–31
- Linux operating system, 37
- MacOS operating system, 37
- MKS Toolkit, 34–35
- Oracle Solaris operating system, 36

- standalone configurations, 117

- start-instance subcommand, 104

- start-local-instance subcommand, 105

starting

- instances in clusters

- centrally, 87–88

- instances individually

- centrally, 89–90

- locally, 98–99

- SSH server daemon

- Cygwin, 30–31

- Linux operating system, 37

- MacOS operating system, 37

- MKS Toolkit, 34–35

- Oracle Solaris operating system, 36

- startup files, 53

- stateful session beans, 147

- session persistence, 147

- session persistence of, 148

stopping

- instances in clusters

- centrally, 88–89

- instances individually

- centrally, 90–91

- locally, 99–100

- StrictModes option

- Cygwin, 31

- Linux operating system, 37

- Oracle Solaris operating system, 36

- sun-ejb-jar.xml file, 149

- synchronization, named configurations, 118–119

- system properties, instances, 126–128

T

- targets, listing for configurations, 121–122

testing

- nodes

- SSH, 54–55

- SSH setup, 38–39

transactions

- and session persistence, 147, 149

U

- unencrypted public key authentication, 40–42

- uninstalling, GlassFish Server software from multiple

- hosts, 48–49

- UNIX operating system, obtaining SSH software

- for, 27

updating

- nodes

- CONFIG, 61

- SSH, 60–61

- upgrade, rolling, 133

upgrading

- nodes

- CONFIG nodes to SSH nodes, 62–63

- userinfo command

- MKS Toolkit, 33, 34

- users, SSH, 27–28

V

- verify-failure-connect-timeout-in-millis setting, 68

- verify-failure-waittime-in-millis setting, 67

W

- Windows operating system

- home directories

- Cygwin, 30

- MKS Toolkit, 32–34

- HOME environment variable, 34

- HOMEDRIVE environment variable, 33

- HOMEPATH environment variable, 33

Windows operating system (*Continued*)

- obtaining SSH software for, 27

- paths

 - Cygwin, 29

 - MKS Toolkit, 32

- setting up Cygwin on, 28–31

- setting up MKS Toolkit on, 31–35

- SSH server daemon

 - Cygwin, 30–31

 - MKS Toolkit, 34–35