

# Eclipse GlassFish Administration Guide, Release 7

# Eclipse GlassFish

Administration Guide

Release 7, DRAFT

Contributed 2018 - 2022

Eclipse GlassFish 7 Administration Guide provides instructions for configuring and administering Eclipse GlassFish.

---

Eclipse GlassFish Administration Guide, Release 7

Copyright © 2013, 2019 Oracle and/or its affiliates. All rights reserved.

This program and the accompanying materials are made available under the terms of the Eclipse Public License v. 2.0, which is available at <http://www.eclipse.org/legal/epl-2.0>.

SPDX-License-Identifier: EPL-2.0

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.



# Preface

The Eclipse GlassFish Administration Guide provides instructions for configuring and administering Eclipse GlassFish.

This preface contains information about and conventions for the entire Eclipse GlassFish (Eclipse GlassFish) documentation set.

Eclipse GlassFish 7 is developed through the GlassFish project open-source community at <https://github.com/eclipse-ee4j/glassfish>. The GlassFish project provides a structured process for developing the Eclipse GlassFish platform that makes the new features of the Jakarta EE platform available faster, while maintaining the most important feature of Jakarta EE: compatibility. It enables Java developers to access the Eclipse GlassFish source code and to contribute to the development of the Eclipse GlassFish.

The following topics are addressed here:

- [Eclipse GlassFish Documentation Set](#)
- [Related Documentation](#)
- [Typographic Conventions](#)
- [Symbol Conventions](#)
- [Default Paths and File Names](#)

## Eclipse GlassFish Documentation Set

The Eclipse GlassFish documentation set describes deployment planning and system installation. For an introduction to Eclipse GlassFish, refer to the books in the order in which they are listed in the following table.

Book Title	Description
<a href="#">Release Notes</a>	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.
<a href="#">Quick Start Guide</a>	Explains how to get started with the Eclipse GlassFish product.
<a href="#">Installation Guide</a>	Explains how to install the software and its components.
<a href="#">Upgrade Guide</a>	Explains how to upgrade to the latest version of Eclipse GlassFish. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
<a href="#">Deployment Planning Guide</a>	Explains how to build a production deployment of Eclipse GlassFish that meets the requirements of your system and enterprise.

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

Book Title	Description
<a href="#">Message Queue Developer's Guide for JMX Clients</a>	Describes the application programming interface in Message Queue for programmatically configuring and monitoring Message Queue resources in conformance with the Java Management Extensions (JMX).
<a href="#">Message Queue Developer's Guide for Java Clients</a>	Provides information about concepts and procedures for developing Java messaging applications (Java clients) that work with Eclipse GlassFish.
<a href="#">Message Queue Developer's Guide for C Clients</a>	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.

## Related Documentation

The following tutorials explain how to develop Jakarta EE applications:

- [Your First Cup: An Introduction to the Jakarta EE Platform](#). For beginning Jakarta EE programmers, this short tutorial explains the entire process for developing a simple enterprise application. The sample application is a web application that consists of a component that is based on the Enterprise JavaBeans specification, a JAX-RS web service, and a JavaServer Faces component for the web front end.
- [The Jakarta EE Tutorial](#). This comprehensive tutorial explains how to use Jakarta EE platform technologies and APIs to develop Jakarta EE applications.

Javadoc tool reference documentation for packages that are provided with Eclipse GlassFish is available as follows.

- The Jakarta EE specifications and API specification is located at <https://jakarta.ee/specifications/>.
- The API specification for Eclipse GlassFish 7, including Jakarta EE platform packages and nonplatform packages that are specific to the Eclipse GlassFish product, is located at <https://glassfish.org/docs/>.

For information about creating enterprise applications in the NetBeans Integrated Development Environment (IDE), see the [NetBeans Documentation, Training & Support page](#).

For information about the Derby database for use with the Eclipse GlassFish, see the [Derby page](#).

The Jakarta EE Samples project is a collection of sample applications that demonstrate a broad range of Jakarta EE technologies. The Jakarta EE Samples are bundled with the Jakarta EE Software Development Kit (SDK) and are also available from the repository (<https://github.com/eclipse-ee4j/glassfish-samples>).

# Typographic Conventions

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

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% you have mail.</code>
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name% su</code>  <code>Password:</code>
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is <code>rm</code> filename.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the User’s Guide.  A cache is a copy that is stored locally.  Do not save the file.

# Symbol Conventions

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

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.
>	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

# Default Paths and File Names

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

Placeholder	Description	Default Value
as-install	Represents the base installation directory for Eclipse GlassFish. In configuration files, as-install is represented as follows: <code> \${com.sun.aas.installRoot}</code>	<ul style="list-style-type: none"><li>Installations on the Oracle Solaris operating system, Linux operating system, and Mac OS operating system: user's-home-directory/<code>glassfish7/glassfish</code></li><li>Installations on the Windows operating system: SystemDrive:<code>\glassfish7\glassfish</code></li></ul>
as-install-parent	Represents the parent of the base installation directory for Eclipse GlassFish.	<ul style="list-style-type: none"><li>Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: user's-home-directory/<code>glassfish7</code></li><li>Installations on the Windows operating system: SystemDrive:<code>\glassfish7</code></li></ul>
domain-root-dir	Represents the directory in which a domain is created by default.	as-install/ <code>domains/</code>
domain-dir	Represents the directory in which a domain's configuration is stored. In configuration files, domain-dir is represented as follows: <code> \${com.sun.aas.instanceRoot}</code>	domain-root-dir/domain-name
instance-dir	Represents the directory for a server instance.	domain-dir/instance-name

# 1 Overview of Eclipse GlassFish Administration

Eclipse GlassFish provides a server for developing and deploying Java Platform Enterprise Edition (Jakarta EE) applications and web Java Web Services.

As an administrator of Eclipse GlassFish, your main responsibilities are to establish a secure Eclipse GlassFish environment and to oversee the services, resources, and users that participate in that environment. Your key tasks include configuring resources and services, managing Eclipse GlassFish at runtime, and fixing problems that are associated with the server. You might also be involved in installing software, integrating add-on components, and deploying applications.

The following topics are addressed here:

- [Default Settings and Locations](#)
- [Configuration Tasks](#)
- [Administration Tools](#)
- [Instructions for Administering Eclipse GlassFish](#)

## Default Settings and Locations

After installation, you might need to perform some immediate configuration tasks to make your installation function as intended. If configuration defaults have been accepted, some features are enabled and some not. For an overview of initial configuration tasks for Eclipse GlassFish services and resources, see [Initial Configuration Tasks](#).

In addition, you might want to reset default passwords, change names or locations of files, and so on. The following tables list the default administration values.



For the zip bundle of Eclipse GlassFish 7, the default administrator login is `admin`, with no password, which means that no login is required.

Table 1-1 Default Administration Values

Item	Default
Domain Name	<code>domain1</code>
Master Password	<code>changeit</code>
Administration User	<code>admin</code>
Administration Server Port	4848
HTTP Port	8080

<b>Item</b>	<b>Default</b>
HTTPS Port	8181
Pure JMX Clients Port	8686
Message Queue Port	7676
IIOP Port	3700
IIOP/SSL Port	3820
IIOP/SSL Port With Mutual Authentication	3920

Table 1-2 Default Locations

<b>Item</b>	<b>Default</b>
Command-line Utility ( <code>asadmin</code> )	<code>as-install/bin</code>
Configuration Files	<code>domain-dir/config</code>
Log Files	<code>domain-dir/logs</code>
Upgrade Tool ( <code>asupgrade</code> Command)	<code>as-install/bin</code>

For information about replaceable items and default paths and files, see [Default Paths and File Names](#).

## Configuration Tasks

Some configuration tasks must be performed directly after installation for your Eclipse GlassFish environment to work as intended. For example, if you are using a database with Eclipse GlassFish, you need to set up database connectivity right away.

Some configuration situations are ongoing and will require you to make changes many times during the life of your installation. You can use either the Administration Console or the `asadmin` utility to modify the configuration. Changes are automatically applied to the appropriate configuration file.

The following topics are addressed here:

- [Initial Configuration Tasks](#)
- [How Dotted Names Work for Configuration](#)
- [Configuration Files](#)
- [Impact of Configuration Changes](#)

## Initial Configuration Tasks

This section maps the common configuration tasks to the command-line procedures in this guide. In some situations, the resource or service is automatically enabled and your configuration tasks involve adjusting or changing the default settings to suit your specific needs.

The following resources and services frequently require configuration immediately after installation:

### System Properties

See [Administering System Properties](#).

### Domains

The initial `domain1` is created during installation. Additional configuration tasks might include such tasks as configuring additional domains or setting up automatic restart. See [Administering Domains](#).

### JVM

The initial tasks for configuring the JVM include creating JVM options and profilers. See [Administering the Virtual Machine for the Java Platform](#).

### Logging

By default, logging is enabled, so basic logging works without additional configuration. However, you might want to change log levels, property values, or the location of log files. See [Administering the Logging Service](#).

### Monitoring

By default, the monitoring service is enabled. However, monitoring for the individual modules is not enabled, so your first monitoring task is to enable monitoring for the modules that you want to monitor. See [Administering the Monitoring Service](#).

### Life Cycle Modules

See [Administering Life Cycle Modules](#).

### Security

- System Security. Initial configuration tasks might include setting up passwords, audit modules, and certificates. See "Administering System Security" in Eclipse GlassFish Security Guide.
- User Security. Initial configuration tasks might include creating authentication realms and file users. See "Administering User Security" in Eclipse GlassFish Security Guide.
- Message Security. Initial configuration tasks might include configuring a Java Cryptography Extension (JCE) provider, enabling default and non-default security providers, and configuring message protection policies. See "Administering Message Security" in Eclipse GlassFish Security Guide.

### Database Connectivity

The initial tasks involved in configuring Eclipse GlassFish to connect to the Apache Derby database include creating a Java Database Connectivity (JDBC) connection pool, creating a JDBC

resource, and integrating a JDBC driver. See [Administering Database Connectivity](#).

## EIS Connectivity

The initial tasks involved in configuring Eclipse GlassFish to connect to an enterprise information system (EIS) include creating a connector connection pool, creating a connector resource, editing a resource adapter configuration, creating a connector security map, creating a connector work security map, and creating an administered object (if needed). See [Administering EIS Connectivity](#).

## Internet Connectivity

The initial tasks involved in making deployed web applications accessible by internet clients include creating HTTP network listeners and virtual servers, and configuring the HTTP listeners for SSL (if needed). See [Administering Internet Connectivity](#).

## Object Request Broker (ORB)

An initial configuration task might involve creating an IIOP listener. See [Administering the Object Request Broker \(ORB\)](#).

## JavaMail Service

An initial configuration task might involve creating a JavaMail resource. See [Administering the JavaMail Service](#).

## Java Message Service (JMS)

Initial configuration tasks might include creating a physical destination, creating connection factories or destination resources, creating a JMS host (if the default JMS host is not adequate), adjusting connection pool settings (if needed), and configuring resource adapters for JMS. See [Administering the Java Message Service \(JMS\)](#).

## JNDI Service

An initial configuration task might involve creating a JNDI resource. See [Administering the Java Naming and Directory Interface \(JNDI\) Service](#).

Information and instructions for accomplishing the tasks by using the Administration Console are contained in the Administration Console online help.

## How Dotted Names Work for Configuration

After the initial configuration is working, you will continue to manage ongoing configuration for the life of your Eclipse GlassFish installation. You might need to adjust resources to improve productivity, or issues might arise that require settings to be modified or defaults to be reset. In some situations, an `asadmin` subcommand is provided for updating, such as the `update-connector-work-security-map` subcommand. However, most updating is done by using the `list`, `get`, and `set` subcommands with dotted names. For detailed information about dotted names, see the [dotted-names\(5ASC\)](#) help page.



Dotted names also apply to monitoring, but the method is different. For

information on using dotted names for monitoring, see [How the Monitoring Tree Structure Works](#).

The general process for working with configuration changes on the command line is as follows:

1. List the modules for the component of interest.

The following single mode example uses the | (pipe) character and the `grep` command to narrow the search:

```
asadmin list "*" | grep http | grep listener
```

Information similar to the following is returned:

```
configs.config.server-config.network-config.network-listeners.network-
listener.http-listener-1
configs.config.server-config.network-config.network-listeners.network-
listener.http-listener-2
configs.config.server-config.network-config.protocols.protocol.admin-listener.http
configs.config.server-config.network-config.protocols.protocol.admin-
listener.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-1
configs.config.server-config.network-config.protocols.protocol.http-listener-1.http
configs.config.server-config.network-config.protocols.protocol.http-listener-
1.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-2
configs.config.server-config.network-config.protocols.protocol.http-listener-2.http
configs.config.server-config.network-config.protocols.protocol.http-listener-
2.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-2.ssl
```

2. Get the attributes that apply to the module you are interested in.

The following multimode example gets the attributes and values for `http-listener-1`:

```
asadmin> get server-config.network-config.network-listeners.network-listener.http-
listener-1.*
```

Information similar to the following is returned:

```
server.http-service.http-listener.http-listener-1.acceptor-threads = 1
server.http-service.http-listener.http-listener-1.address = 0.0.0.0
server.http-service.http-listener.http-listener-1.blocking-enabled = false
server.http-service.http-listener.http-listener-1.default-virtual-server = server
server.http-service.http-listener.http-listener-1.enabled = true
server.http-service.http-listener.http-listener-1.external-port =
server.http-service.http-listener.http-listener-1.family = inet
```

```
server.http-service.http-listener.http-listener-1.id = http-listener-1
server.http-service.http-listener.http-listener-1.port = 8080
server.http-service.http-listener.http-listener-1.redirect-port =
server.http-service.http-listener.http-listener-1.security-enabled = false
server.http-service.http-listener.http-listener-1.server-name =
server.http-service.http-listener.http-listener-1.xpowered-by = true
```

3. Modify an attribute by using the `set` subcommand.

This example sets the `security-enabled` attribute of `http-listener-1` to true:

```
asadmin> set server.http-service.http-listener.http-listener-1.security-enabled =
true
```

## Configuration Files

The bulk of the configuration information about Eclipse GlassFish resources, applications, and instances is stored in the `domain.xml` configuration file. This file is the central repository for a given administrative domain and contains an XML representation of the Eclipse GlassFish domain model. The default location for the `domain.xml` file is `domain-dir/config`.

 Eclipse GlassFish maintains a backup of the `domain.xml` file that is named `domain.xml.bak`. The purpose of this file is solely to enable Eclipse GlassFish to start a domain if the `domain.xml` file cannot be read. Do not modify or delete the `domain.xml.bak` file and do not use this file for any other purpose.

The `logging.properties` file is used to configure the Java Util Logging system. The default `logging.properties` file is located in the same directory as the `domain.xml` file. For further information on the `logging.properties` file, see [Logging Properties](#).

The `asenv.conf` file is located in the `as-install/config` directory. Its purpose is to store the Eclipse GlassFish environment variables, such as the installation location of the database, Message Queue, and so on.

 Changes are automatically applied to the appropriate configuration file. Do not edit the configuration files directly. Manual editing is prone to error and can have unexpected results.

## Impact of Configuration Changes

Some configuration changes require that you restart the DAS or Eclipse GlassFish instances for the changes to take effect. Other changes are applied dynamically without requiring that the DAS or instances be restarted. The procedures in this guide indicate when a restart is required. Eclipse

GlassFish enables you to determine whether the DAS or an instance must be restarted to apply configuration changes.

Some changes to resources or connection pools affect the applications that use the resources or connection pools. These changes do not require restart. However, any applications that use the resources or connection pools must be disabled and re-enabled or redeployed for the change to take effect.

The following topics are addressed here:

- [To Determine Whether the DAS or an Instance Requires Restart](#)
- [Configuration Changes That Require Restart](#)
- [Dynamic Configuration Changes](#)
- [Changes That Affect Applications](#)

## To Determine Whether the DAS or an Instance Requires Restart

1. Ensure that the DAS is running. To obtain information about the DAS or an instance, a running server is required.
2. Do one of the following:
  - To determine if the DAS requires restart, list the domains in your Eclipse GlassFish installation. Use the `list-domains` subcommand for this purpose.

```
asadmin> list-domains [--domaindir domain-root-dir]
```

The domain-root-dir is the directory that contains the directories in which individual domains' configuration is stored. The default is as-install/`domains`, where as-install is the base installation directory of the Eclipse GlassFish software. If the DAS requires restart, a statement that restart is required is displayed.

- To determine if an instance requires restart, list information about the instance. Use the `list-instances` subcommand for this purpose.

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

The instance-name is the name of the instance for which you are listing information. If the instance requires restart, one of the following pieces of information is displayed: a statement that restart is required, or a list of configuration changes that are not yet applied to the instance.

### Example 1-1 Determining if the DAS Requires Restart

This example determines that the DAS for the domain `domain1` requires restart to apply configuration changes.

```
asadmin> list-domains
domain1 running, restart required to apply configuration changes
Command list-domains executed successfully.
```

### Example 1-2 Determining if an Instance Requires Restart

This example determines that the instance `pmd-i1` requires restart to apply configuration changes.

```
asadmin> list-instances pmd-i1
pmd-i1  running;  requires restart
Command list-instances executed successfully.
```

### See Also

- [list-domains\(1\)](#)
- [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-domains](#)
- [asadmin help list-instances](#)

## Configuration Changes That Require Restart

The following configuration changes require restart for the changes to take effect:

- Changing JVM options
- Changing port numbers



Changes to some port numbers, for example HTTP listener ports, do not require restart.

- Changing log handler elements
- Configuring certificates
- Managing HTTP, JMS, IIOP, JNDI services
- Enabling or disabling secure administration as explained in "Running Secure Admin" in Eclipse GlassFish Security Guide

## Dynamic Configuration Changes

With dynamic configuration, changes take effect while the DAS or instance is running. The

following configuration changes do not require restart:

- Adding or deleting add-on components
- Adding or removing JDBC, JMS, and connector resources and pools (Exception: Some connection pool properties affect applications.)
- Changing a system property that is not referenced by a JVM option or a port
- Adding file realm users
- Changing logging levels
- Enabling and disabling monitoring
- Changing monitoring levels for modules
- Enabling and disabling resources and applications
- Deploying, undeploying, and redeploying applications

## Changes That Affect Applications

Some changes to resources or connection pools affect the applications that use the resources or connection pools. These changes do not require restart. However, any applications that use the resources or connection pools must be disabled and re-enabled or redeployed for the change to take effect.



If you do not know which applications use the changed resources or connection pools, you can apply these changes by restarting the clusters or Eclipse GlassFish instances to which applications are deployed. However, to minimize the disruption to the services that your applications provide, avoid restarting clusters or instances to apply these changes if possible.

The following changes affect applications:

- Creating or deleting resources (Exception: Changes to some JDBC, JMS, or connector resources do not affect applications.)
- Modifying the following JDBC connection pool properties:
  - `datasource-classname`
  - `associate-with-thread`
  - `lazy-connection-association`
  - `lazy-connection-enlistment`
  - JDBC driver vendor-specific properties
- Modifying the following connector connection pool properties:
  - `resource-adapter-name`
  - `connection-definition-name`
  - `transaction-support`

- `associate-with-thread`
- `lazy-connection-association`
- `lazy-connection-enlistment`
- Vendor-specific properties

## Administration Tools

For the most part, you can perform the same tasks by using either the graphical Administration Console or the `asadmin` command-line utility, however, there are exceptions.

The following Eclipse GlassFish administration tools are described here:

- [Administration Console](#)
- [asadmin Utility](#)
- [REST Interfaces](#)
- [OSGi Module Management Subsystem](#)
- [keytool Utility](#)
- [Java Monitoring and Management Console \(JConsole\)](#)

## Administration Console

The Administration Console is a browser-based utility that features an easy-to-navigate graphical interface that includes extensive online help for the administrative tasks.

To use the Administration Console, the domain administration server (DAS) must be running. Each domain has its own DAS, which has a unique port number. When Eclipse GlassFish was installed, you chose a port number for the DAS, or used the default port of 4848. You also specified a user name and password if you did not accept the default login (`admin` with no password).

When specifying the URL for the Administration Console, use the port number for the domain to be administered. The format for starting the Administration Console in a web browser is `http://`hostname`:`port`. For example:

```
http://kindness.example.com:4848
```

If the Administration Console is running on the host where Eclipse GlassFish was installed, specify `localhost` for the host name. For example:

```
http://localhost:4848
```

If the Administration Console is run on a host different from the host where Eclipse GlassFish was installed, a secure connection ([https](https://) instead of [http](http://)) is used. Some browsers do not display pages on secure connections by default and must be configured to permit secure protocols (SSL and TLS).

For Microsoft Windows, an alternate way to start the Eclipse GlassFish Administration Console is by using the Start menu.

You can display the help material for a page in the Administration Console by clicking the Help button on the page. The initial help page describes the functions and fields of the page itself. Associated task instructions can be accessed on additional pages by clicking a link in the See Also list.

If you try to use the Administration Console from a system through a proxy server on another system back to the original system, while using the system's full host name (instead of <localhost> or <127.0.0.1>) you are denied access because the request is treated as a remote request, which requires that the secure administration feature (secure admin) be enabled.

To avoid this situation, do one of the following:



- Do not use a proxy server.
- Use <localhost> or <127.0.0.1> as the host name.
- Enable secure admin so that what Eclipse GlassFish interprets as a remote request is accepted as such.

To enable secure admin, see "[Managing Administrative Security](#)" in Eclipse GlassFish Security Guide.

## asadmin Utility

The <asadmin> utility is a command-line tool that runs subcommands for identifying the operation or task that you want to perform. You can run <asadmin> subcommands either from a command prompt or from a script. Running <asadmin> subcommands from a script is helpful for automating repetitive tasks. Basic information about how the <asadmin> utility works can be found in the [asadmin\(1M\)](#) help page. For instructions on using the <asadmin> utility, see [Using the asadmin Utility](#).

To issue an <asadmin> subcommand in the standard command shell (single mode), go to the `as-install/bin` directory and type the <asadmin> command followed by a subcommand. For example:

```
asadmin list-jdbc-resources
```

You can invoke multiple command mode (multimode) by typing <asadmin> at the command prompt, after which the <asadmin>> prompt is presented. The <asadmin> utility continues to accept subcommands until you exit multimode and return to the standard command shell. For example:

```
asadmin> list-jdbc-resources
```

You can display a help page for any `asadmin` subcommand by typing `help` before the subcommand name. For example:

```
asadmin> help restart-domain
```

or

```
asadmin help restart-domain
```

A collection of the `asadmin` help pages is available in HTML and PDF format in the [Eclipse GlassFish Reference Manual](#).

## REST Interfaces

Eclipse GlassFish provides representational state transfer (REST) interfaces to enable you to access monitoring and configuration data for Eclipse GlassFish, including data that is provided by newly installed add-on components. For more information, see [Using REST Interfaces to Administer Eclipse GlassFish](#).

## OSGi Module Management Subsystem

The OSGi module management subsystem that is provided with Eclipse GlassFish is the [Apache Felix OSGi framework](#). To administer this framework, use the either of the following tools:

- [Apache Felix Gogo](#) remote shell. This shell is provided with Eclipse GlassFish. The shell uses the Felix Gogo shell service to interact with the OSGi module management subsystem.
- GlassFish OSGi Administration Console. This console is distributed as an add-on component for Eclipse GlassFish or as a set of files from the Maven GlassFish repository. In both distributions, the GlassFish OSGi Web Console is provided as an extension to the Administration Console and as a standalone web application. The GlassFish OSGi Administration Console is a customized version of the [Apache Felix Web Console](#).

These tools enable you to perform administrative tasks on OSGi bundles such as:

- Browsing installed OSGi bundles
- Viewing the headers of installed OSGi bundles
- Installing OSGi bundles
- Controlling the life cycle of installed bundles

## To Enable the Apache Felix Gogo Remote Shell

By default, the Apache Felix Gogo remote shell in Eclipse GlassFish is disabled. Before using the shell to administer OSGi bundles in Eclipse GlassFish, you must enable the shell.

Enabling the Apache Felix Gogo remote shell in Eclipse GlassFish involves changing the value of the property `glassfish.osgi.start.level.final`. This property controls whether the OSGi start level service enables the shell when the DAS or a Eclipse GlassFish instance is started.

1. Ensure that the DAS is running.
2. Change the value of the `glassfish.osgi.start.level.final` property from 2 to 3. If the domain includes clustered or standalone instances on remote hosts, perform this step on each remote host. You can change this value either by creating a Java system property or by editing a file.
  - To change this value by creating a Java system property, create the Java system property `glassfish.osgi.start.level.final` with a value of 3.

```
asadmin> create-jvm-options --target target -Dglassfish.osgi.start.level.final=3
```

### target

The target for which you are creating the property.

For the DAS, the target is 'server'.

For a clustered or standalone instance, the target is the name of the instance.

\* To change this value by editing a file, edit the plain-text file `as-install``/config/osgi.properties``` to change the value of the '`glassfish.osgi.start.level.final`' property from 2 to 3.

3. Restart the DAS. For instructions, see [To Restart a Domain](#).

## To Run Apache Felix Gogo Remote Shell Commands

The Apache Felix Gogo remote shell is integrated with the Eclipse GlassFish `asadmin` command line utility. You can use the `asadmin` subcommands `osgi` and `osgi-shell` to access the remote shell and run OSGi shell commands.

### To Run Remote Shell Commands Using the `osgi` Subcommand

The `osgi` subcommand delegates the command line to the Apache Felix Gogo remote shell for the execution of OSGi shell commands. Commands are executed by the remote shell and results are returned by the `asadmin` utility. The `osgi` subcommand is supported in remote mode only.

1. Ensure that the server is running. Remote commands require a running server.
2. Access the remote shell by using the `osgi` subcommand. For the full syntax and options for this subcommand, see [osgi\(1\)](#).

#### To Run Remote Shell Commands Using the `osgi-shell` Subcommand

The `osgi-shell` subcommand provides interactive access to the Apache Felix Gogo remote shell for the execution of OSGi shell commands. OSGi shell commands are executed on the server and results are printed on the client. You can run multiple commands from a file or run commands interactively. The `osgi-shell` subcommand is supported in local mode only. Unlike other local subcommands, however, the DAS and the server instance whose shell is being accessed must be running.

1. Ensure that the server is running.
2. Access the remote shell by using the `osgi-shell` subcommand. For the full syntax and options for this subcommand, see [osgi-shell\(1\)](#).

#### Example 1-3 Listing Apache Felix Gogo Remote Shell Commands

This example lists Apache Felix Gogo remote shell commands. Some lines of output are omitted from this example for readability.

```
asadmin> osgi help
felix:bundlelevel
felix:cd
felix:frameworklevel
gogo:cat
gogo:each
gogo:echo
...
asadmin> osgi-shell
Use "exit" to exit and "help" for online help.
gogo$ help
felix:bundlelevel
felix:cd
felix:frameworklevel
gogo:cat
gogo:each
gogo:echo
```

#### Example 1-4 Running a Remote Shell Command

This example runs the Felix Remote Shell Command `lb` without any arguments to list all installed OSGi bundles. Some lines of output are omitted from this example for readability.

```
asadmin> osgi lb
START LEVEL 2
ID|State      |Level|Name
```

```

0|Active | 0|System Bundle
1|Active | 1|Metro Web Services API OSGi Bundle
2|Active | 1|jakarta.annotation API
Command osgi executed successfully.

...
asadmin> osgi-shell
Use "exit" to exit and "help" for online help.
gogo$ lb
START LEVEL 2
ID|State |Level|Name
0|Active | 0|System Bundle
1|Active | 1|Metro Web Services API OSGi Bundle
2|Active | 1|jakarta.annotation API
gogo$
```

#### Example 1-5 Determining the Services That an OSGi Bundle Provides

This example runs the Felix Remote Shell Command `inspect` with the `service` option and the `capability` option to determine the services that OSGi bundle 251 provides. Some lines of output are omitted from this example for readability.

```

asadmin> osgi inspect service capability 251
== GlassFish EJB Container for OSGi Enabled EJB Applications (251) provides services:
objectClass = org.glassfish.osgijavaeebase.Extender
service.id = 68
-----
objectClass = org.glassfish.osgijavaeebase.OSGiDeployer
service.id = 69
service.ranking = -2147483648
Command osgi executed successfully.

...
asadmin> osgi -shell
Use "exit" to exit and "help" for online help.
gogo$ inspect service capability 251
== GlassFish EJB Container for OSGi Enabled EJB Applications (251) provides services:
objectClass = org.glassfish.osgijavaeebase.Extender
service.id = 68
...
gogo$
```

#### To Download and Install the GlassFish OSGi Web Console

The GlassFish OSGi Web Console is distributed as follows:

- As an add-on component for Eclipse GlassFish
- As a set of files from the [GlassFish Maven repository](#)

In both distributions, the GlassFish OSGi Web Console is provided as an extension to the Administration Console and as a standalone web application.

1. Perform one of the following sets of steps, depending on how you are obtaining the GlassFish OSGi Web Console.
  - If you are obtaining the console as an add-on component, install the GlassFish OSGi Admin Console component.
  - If you are obtaining the console from the Maven repository, download and unzip the required files.
2. Download the following files to the parent of the [glassfish7](#) directory of your Eclipse GlassFish installation. [glassfish-osgi-http-3.1.2.zip](#)  
[glassfish-osgi-gui-3.1.2.zip](#)
3. Unzip the files that you downloaded.

The contents of the files are added to the as-install/[modules/autostart](#) directory of your Eclipse GlassFish installation.

4. Restart the DAS. For instructions, see [To Restart a Domain](#).

## Next Steps

After downloading and installing the GlassFish OSGi Web Console, you can access the console as explained in the following sections:

- [To Access the GlassFish OSGi Web Console Through the Eclipse GlassFish Administration Console](#)
- [To Access the GlassFish OSGi Web Console as a Standalone Web Application](#)

## To Access the GlassFish OSGi Web Console Through the Eclipse GlassFish Administration Console

A tab for the GlassFish OSGi Web Console is provided for the DAS and for every Eclipse GlassFish instance in a domain.

1. Ensure that the DAS and the instance for which you want to access the GlassFish OSGi Web Console are running.
2. Start the Eclipse GlassFish Administration Console. For instructions, see [Administration Console](#).
3. Open the Administration Console page for the DAS or instance for which you are accessing the GlassFish OSGi Web Console.
  - For the DAS, in the navigation tree, select the server (Admin Server) node.
  - For a standalone instance, perform these steps:
    - a. In the navigation tree, expand the Standalone Instances node.
    - b. Under the Standalone Instances node, select the instance.

- For a clustered instance, perform these steps:
  - a. In the navigation tree, expand the Clusters node.
  - b. Under the Clusters node, select the cluster that contains the instance. The General Information page for the cluster opens.
  - c. In the General Information page for the cluster, click the Instances tab. The Clustered Server Instances page for the cluster opens.
  - d. In the Server Instances table on the Clustered Server Instances page, select the instance.
- 4. On the Administration Console page for the DAS or instance, click the OSGi Console tab. You are prompted for the user name and password of the administrative user of the GlassFish OSGi Web Console.
- 5. In response to the prompt, provide the user name and password of the administrative user of the GlassFish OSGi Web Console. The user name and password of this user are both preset to **admin**. The GlassFish OSGi Web Console page opens.

## To Access the GlassFish OSGi Web Console as a Standalone Web Application

1. Ensure that the DAS or the instance for which you want to access the GlassFish OSGi Web Console is running.
2. In a web browser, open the following location:

```
http://host:http-port/osgi/system/console/
```

### host

The host where the DAS or instance is running.

### http-port

The port on which Eclipse GlassFish listens for HTTP requests. The default is 8080.

For example, if the DAS is running on the local host and Eclipse GlassFish listens for HTTP requests on the default port, open the following location:

+

```
http://localhost:8080/osgi/system/console/
```

3. When prompted, provide the user name and password of the administrative user of the GlassFish OSGi Web Console.

The user name and password of this user are both preset to **admin**.

## keytool Utility

The `keytool` utility is used to set up and work with Java Security Socket Extension (JSSE) digital certificates. See "[Administering JSSE Certificates](#)" in Eclipse GlassFish Security Guide for instructions on using `keytool`.

## Java Monitoring and Management Console (JConsole)

Java SE provides tools to connect to an MBean server and view the MBeans that are registered with the server. JConsole is one such popular JMX Connector Client and is available as part of the standard Java SE distribution. For instructions on implementing JConsole in the Eclipse GlassFish environment, see [Configuring JConsole to View Eclipse GlassFish Monitoring Data](#).

# Instructions for Administering Eclipse GlassFish

Information and instructions on performing most of the administration tasks from the command line are provided in this document and in the `asadmin` utility help pages. For instructions on accessing `asadmin` online help, see [To Display Help Information for the asadmin Utility or a Subcommand](#).

Information and instructions for accomplishing the tasks by using the Administration Console are contained in the Administration Console online help.



Instructions written for the Eclipse GlassFish tools use standard UNIX forward slashes (/) for directory path separators in commands and file names. If you are running Eclipse GlassFish on a Microsoft Windows system, use backslashes (\) instead. For example:

- UNIX: `as-install/bin/asadmin`
- Windows: `as-install\bin\asadmin`

The following additional documents address specific administration areas:

- Verifying and deploying applications [Eclipse GlassFish Application Deployment Guide](#)

# **Part I**

# **Runtime Administration**

# 2 General Administration

This chapter provides instructions for performing general administration tasks in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [Using the `asadmin` Utility](#)
- [Administering System Properties](#)
- [Using Configuration Modularity](#)
- [Administering Resources](#)
- [Listing Various System Elements](#)
- [Using REST Interfaces to Administer Eclipse GlassFish](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## Using the `asadmin` Utility

Use the `asadmin` utility to perform administrative tasks for Eclipse GlassFish from the command line or from a script. You can use this utility instead of the Administration Console interface.

The following topics are addressed here:

- [Path to the `asadmin` Utility](#)
- [`asadmin` Utility Syntax](#)
- [To Run an `asadmin` Utility Subcommand in Single Mode](#)
- [To Display Help Information for the `asadmin` Utility or a Subcommand](#)
- [To Start a Multimode Session](#)
- [To End a Multimode Session](#)
- [To Run a Set of `asadmin` Subcommands From a File](#)
- [To Run `asadmin` Subcommands in `--detach` Mode](#)

### Path to the `asadmin` Utility

The `asadmin` utility is located in the `as-install/bin` directory. To run the `asadmin` utility without specifying the path, ensure that this directory is in your path.

## asadmin Utility Syntax

The syntax for running the `asadmin` utility is as follows:

```
asadmin [asadmin-util-options] [subcommand [subcommand-options] [operands]]
```

The replaceable items in this syntax are described in the subsections that follow. For full details of this syntax, see the [asadmin\(1M\)](#) help page.

### Subcommands of the `asadmin` Utility

The subcommand identifies the operation or task that you are performing. Subcommands are case-sensitive. Each subcommand is either a local subcommand or a remote subcommand.

- A local subcommand can be run without a running domain administration server (DAS). However, to run the subcommand and have access to the installation directory and the domain directory, the user must be logged in to the machine that hosts the domain.
- A remote subcommand is always run by connecting to a DAS and running the subcommand there. A running DAS is required.

For a list of the subcommands for this release of Eclipse GlassFish, see Section 1 of the [Eclipse GlassFish Reference Manual](#).

### asadmin Utility Options and Subcommand Options

Options control the behavior of the `asadmin` utility and its subcommands. Options are case-sensitive.

The `asadmin` utility has the following types of options:

- `asadmin` utility options. These options control the behavior of the `asadmin` utility, not the subcommand. The `asadmin` utility options may precede or follow the subcommand, but `asadmin` utility options after the subcommand are deprecated. All `asadmin` utility options must either precede or follow the subcommand. If `asadmin` utility options are specified both before and after the subcommand, an error occurs. For a description of the `asadmin` utility options, see the [asadmin\(1M\)](#) help page.
- Subcommand Options. These options control the behavior of the subcommand, not the `asadmin` utility. Subcommand options must follow the subcommand. For a description of a subcommand's options, see the entry for the subcommand in the [Eclipse GlassFish Reference Manual](#).



Not all subcommand options are supported for this release of Eclipse GlassFish. If you specify an unsupported option, a syntax error does not occur. Instead, the command runs successfully and the unsupported option is silently ignored.

A subcommand option may have the same name as an `asadmin` utility option, but the effects of the two options are different.

Options have a long form and a short form.

- The short form of an option has a single dash (-) followed by a single character.
- The long form of an option has two dashes (--) followed by an option word.

For example, the short form and the long form of the option for specifying terse output are as follows:

- Short form: `-t`
- Long form: `--terse`

Most options require argument values, except Boolean options, which toggle to enable or disable a feature.

## Operands of `asadmin` Utility Subcommands

Operands specify the items on which the subcommand is to act. Operands must follow the argument values of subcommand options, and are set off by a space, a tab, or double dashes (--). The `asadmin` utility treats anything that follows the subcommand options and their values as an operand.

## To Run an `asadmin` Utility Subcommand in Single Mode

In single mode, you must type a separate `asadmin` command for each subcommand that you want to use. After the subcommand has run, you are returned to the operating system's command shell. Any `asadmin` utility options must be specified in each separate `asadmin` command that you run. If you require the same `asadmin` utility options for multiple subcommands, use the `asadmin` utility in multimode. For more information, see [To Start a Multimode Session](#).

1. In the operating system's command shell, run the `asadmin` utility, specifying the subcommand.
2. If necessary, also specify any required `asadmin` utility options, subcommand options, and operands.

### Example 2-1 Running an `asadmin` Utility Subcommand in Single Mode

This example runs the `list-applications` subcommand in single mode. In this example, the default values for all options are used.

The example shows that the application `hello` is deployed on the local host.

```
asadmin list-applications
hello <web>
```

```
Command list-applications executed successfully.
```

#### Example 2-2 Specifying an `asadmin` Utility Option With a Subcommand in Single Mode

This example specifies the `--host asadmin` utility option with the `list-applications` subcommand in single mode. In this example, the DAS is running on the host `srvr1.example.com`.

The example shows that the applications `basic-ezcomp`, `scrumtoys`, `ejb31-war`, and `automatic-timer-ejb` are deployed on the host `srvr1.example.com`.

```
asadmin --host srvr1.example.com list-applications
basic-ezcomp <web>
scrumtoys <web>
ejb31-war <ejb, web>
automatic-timer-ejb <ejb>
Command list-applications executed successfully.
```

#### Example 2-3 Specifying an `asadmin` Utility Option and a Subcommand Option in Single Mode

This example specifies the `--host asadmin` utility option and the `--type` subcommand option with the `list-applications` subcommand in single mode. In this example, the DAS is running on the host `srvr1.example.com` and applications of type `web` are to be listed.

```
asadmin --host srvr1.example.com list-applications --type web
basic-ezcomp <web>
scrumtoys <web>
ejb31-war <ejb, web>
Command list-applications executed successfully.
```

## To Display Help Information for the `asadmin` Utility or a Subcommand

Eclipse GlassFish provides help information about the syntax, purpose, and options of the `asadmin` utility and its subcommands. This help information is written in the style of UNIX platform man pages. This help information is also available in the [Eclipse GlassFish Reference Manual](#).

1. If you are displaying help information for a remote subcommand, ensure that the server is running.

Remote subcommands require a running server.

2. Specify the subcommand of interest as the operand of the `help` subcommand.

If you run the `help` subcommand without an operand, help information for the `asadmin` utility is displayed.

#### Example 2-4 Displaying Help Information for the `asadmin` Utility

This example displays the help information for the `asadmin` utility.

```
asadmin help
```

#### Example 2-5 Displaying Help Information for an `asadmin` Utility Subcommand

This example displays the help information for the `create-jdbc-resource` subcommand.

```
asadmin help create-jdbc-resource
```

#### See Also

To display the available subcommands, use the `list-commands` subcommand. Local subcommands are displayed before remote subcommands. If the server is not running, only local subcommands are displayed.

## To Start a Multimode Session

The `asadmin` utility can be used in multiple command mode, or multimode. In multimode, you run the `asadmin` utility once to start a multimode session. During the session, the `asadmin` utility continues to accept subcommands until you end the session and return to the operating system's command shell. Any `asadmin` utility options that you set for your multimode session are used for all subsequent subcommands in the session.



Starting a multimode session does not require a running DAS.

1. Do one of the following:
  - Run the `asadmin` utility without a subcommand.
  - Use the `multimode` subcommand.
2. If necessary, also specify any `asadmin` utility options that will apply throughout the multimode session.
3. In a multimode session, the `asadmin>` prompt is displayed on the command line. You can now type `asadmin` subcommands at this prompt to administer Eclipse GlassFish.

#### Example 2-6 Starting a Multimode Session With `asadmin` Utility Options

This example starts a multimode session in which the `asadmin` utility options `--user` and `--passwordfile` are set for the session.

```
asadmin --user admin1 --passwordfile pwd.txt multimode
```

#### Example 2-7 Starting a Multimode Session by Using the `multimode` Subcommand

This example uses the `multimode` subcommand to start a multimode session in which the default `asadmin` utility options are used.

```
asadmin multimode
```

The `asadmin>` prompt is displayed on the command line.

#### Example 2-8 Running a Subcommand in a Multimode Session

This example starts a multimode session and runs the `list-domains` subcommand in the session.

```
asadmin
Enter commands one per "line", ^D to quit
asadmin> list-domains
Name: domain1 Status: Running
Command list-domains executed successfully.
asadmin>
```

#### Starting a Multimode Session From Within an Existing Multimode Session

You can start a multimode session from within an existing session by running the `multimode` subcommand from within the existing session. After you end the second multimode session, you return to your original multimode session.

#### See Also

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

## To End a Multimode Session

At the `asadmin>` prompt, type one of the following commands or key combinations:

- `exit`
- `quit`
- UNIX and Linux systems: Ctrl-D
- Windows systems: Ctrl-Z



Do not type Ctrl-C to end a multimode session. If a domain or Eclipse GlassFish instance is started from the multimode session, typing Ctrl-C kills the domain or instance process.

You are returned to the operating system's command shell and the `asadmin>` prompt is no longer displayed. If the `asadmin>` prompt is still displayed, you might have opened a multimode session within a multimode session. In this situation, repeat this procedure to end the remaining

multimode session.

## To Run a Set of `asadmin` Subcommands From a File

Running a set of `asadmin` subcommands from a file enables you to automate repetitive tasks.

1. Create a plain text file that contains the sequence of subcommands that you want to run.
2. Run the `mymode` subcommand, specifying the file that you created.

If necessary, also specify any `asadmin` utility options that are required to enable subcommands in the file to run.

### Example 2-9 Running a Set of `asadmin` Subcommands From a File

This example contains the following:

- A listing of a file that is named `commands_file.txt`, which contains a sequence of `asadmin` subcommands
- The command to run the subcommands in the file `commands_file.txt`

The `commands_file.txt` file contains the `asadmin` utility subcommands to perform the following sequence of operations:

1. Creating the domain `customdomain`
2. Starting the domain `customdomain`
3. Listing all available subcommands
4. Stopping the domain `customdomain`
5. Deleting the domain `customdomain`

The content of the `commands_file.txt` file is as follows:

```
create-domain --portbase 9000 customdomain
start-domain customdomain
list-commands
stop-domain customdomain
delete-domain customdomain
```

This example runs the sequence of subcommands in the `commands_file.txt` file. Because the `--portbase` option is specified for the `create-domain` subcommand in the file, the `--port` `asadmin` utility option must also be set.

```
asadmin --port 9048 multimode --file commands_file.txt
```

See Also

For more information about the subcommands in the preceding example, see the following help pages:

- [create-domain\(1\)](#)
- [delete-domain\(1\)](#)
- [list-commands\(1\)](#)
- [multimode\(1\)](#)
- [start-domain\(1\)](#)
- [stop-domain\(1\)](#)

## To Run `asadmin` Subcommands in `--detach` Mode

You can use the `--detach` option of the `asadmin` utility to detach `asadmin` subcommands and run them in the background in detach mode. The `asadmin --detach` option is useful for long-running subcommands and enables you to run several independent subcommands from one console or script.

1. Ensure that the server is running. Remote commands require a running server.
2. Detach and run the subcommand by using the `asadmin --detach` option.

### Example 2-10 Using the `--detach` Option in Single Mode

This example uses the `asadmin --detach` option in single mode to run the `create-cluster` subcommand.

```
asadmin --detach create-cluster Cluster1
Job ID: 1
Command create-cluster started successfully.
```

### Example 2-11 Using the `--detach` Option in Multimode

This example uses the `asadmin --detach` option in multimode to run the `create-cluster` subcommand.

```
asadmin> create-cluster Cluster1 --detach
Job ID: 1
Command create-cluster started successfully.
```

Job IDs are assigned to subcommands that are started using the `asadmin --detach` option or that contain progress information. You can use the `list-jobs` subcommand to list jobs and their job IDs, the `attach` subcommand to reattach to a job and view its status, and the `configure-managed-jobs` subcommand to configure how long information about jobs is kept.

### Example 2-12 Listing Jobs

This example runs the [list-jobs](#) subcommand in multimode to list jobs and job information.

```
asadmin> list-jobs
JOB ID      COMMAND          STATE      EXIT CODE TIME OF COMPLETION
1           create-cluster   COMPLETED  SUCCESS   2013-02-15 16:16:16 PST
2           deploy           COMPLETED  FAILURE   2013-02-15 18:26:30 PST
Command list-jobs executed successfully
```

#### Example 2-13 Attaching to a Subcommand and Checking Its Status

This example runs the [attach](#) subcommand in multimode to attach to the [create-cluster](#) subcommand with a job ID of 1. If a subcommand is still in progress, the output displays the current status, such as percentage complete.

```
asadmin> attach 1
Command create-cluster executed with status SUCCESS.
Command attach executed successfully.
```

#### Example 2-14 Configuring Managed Jobs

This example runs the [configure-managed-jobs](#) subcommand in multimode to set the job retention period to 36 hours. Time periods can be specified in Hh|Mm|Ss for hours, minutes, or seconds.

```
asadmin> configure-managed-jobs --job-retention-period=36h
Command configure-managed-jobs executed successfully.
```

#### See Also

For the full syntax and options of the subcommands in the preceding examples, see the following help pages:

- [attach\(1\)](#)
- [configure-managed-jobs\(1\)](#)
- [list-jobs\(1\)](#)

## Administering System Properties

Shared server instances will often need to override attributes defined in their referenced configuration. Any configuration attribute can be overridden through a system property of the corresponding name.

The following topics are addressed here:

- [To Create System Properties](#)

- [To List System Properties](#)
- [To Delete a System Property](#)

## To Create System Properties

Use the `create-system-properties` subcommand in remote mode to create or update one or more system properties of the domain or configuration. Any configuration attribute can be overwritten through a system property of the corresponding name.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create system properties by using the `create-system-properties` subcommand.

Information about properties for the subcommand is included in this help page.

### Example 2-15 Creating a System Property

This example creates a system property associated with `http-listener-port=1088` on `localhost`.

```
asadmin> create-system-properties http-listener-port=1088
Command create-system-properties executed successfully.
```

### See Also

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

## To List System Properties

Use the `list-system-properties` subcommand in remote mode to list the system properties that apply to a domain, cluster, or server instance or configuration.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List system properties by using the `list-system-properties` subcommand.

The existing system properties are displayed, including predefined properties such as `HTTP_LISTENER_PORT` and `HTTP_SSL_LISTENER_PORT`.

### Example 2-16 Listing System Properties

This example lists the system properties on host `localhost`.

```
asadmin> list-system-properties
http-listener-port=1088
Command list-system-properties executed successfully.
```

## See Also

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

## To Delete a System Property

Use the `delete-system-property` subcommand in remote mode to delete system properties.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing system properties by using the `list-system-properties` subcommand.
3. Delete the system property by using the `delete-system-property` subcommand.
4. If necessary, notify users that the system property has been deleted.

### Example 2-17 Deleting a System Property

This example deletes a system property named `http-listener-port` from `localhost`.

```
asadmin> delete-system-property http-listener-port
Command delete-system-property executed successfully.
```

## See Also

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

## Using Configuration Modularity

With configuration modularity in Eclipse GlassFish, new modules can be added to Eclipse GlassFish distributions without modifying the global `domain.xml` configuration file. Default configuration data for modules is stored in the modules themselves, rather than in `domain.xml`, and loaded when needed.

Module configuration elements are stored in `domain.xml` only when the default configuration included in the module is changed or when module configuration elements are added to `domain.xml` using the `create-module-config` subcommand. The `delete-module-config` subcommand removes module configuration elements from `domain.xml`, and the `get-active-module-config` subcommand displays the current active configuration of a module.

## To Add the Default Configuration of a Module to `domain.xml`

Use the `create-module-config` subcommand to add the default configuration of a module to

`domain.xml`.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Add the default configuration of a module to `domain.xml` by using the `create-module-config` subcommand.

#### Example 2-18 Adding Module Configuration to `domain.xml`

This example adds the default configuration of the web container module to `domain1` in `server-config` (the default configuration). Use the `--dryrun` option to preview the configuration before it is added.

```
asadmin> create-module-config web-container
Command create-module-config executed successfully.
```

#### See Also

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

## To Remove the Configuration of a Module From `domain.xml`

Use the `delete-module-config` subcommand to remove the configuration of a module from `domain.xml` and cause the module to use the default configuration included in the module.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Remove the configuration of a module from `domain.xml` by using the `delete-module-config` subcommand.

#### Example 2-19 Removing Module Configuration From `domain.xml`

This example deletes the configuration of the web container module from `domain1` in `server-config` (the default configuration).

```
asadmin> delete-module-config web-container
Command delete-module-config executed successfully.
```

#### See Also

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

## To Display the Current Active Configuration of a Module

Use the `get-active-module-config` subcommand to display the current active configuration of a module.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Display the current active configuration of a module by using the `get-active-module-config` subcommand.

### Example 2-20 Displaying the Current Active Configuration of a Module

This example displays the current active configuration of the JMS service in `server-config` (the default configuration).

```
asadmin> get-active-module-config jms-service
At location: domain/configs/config[server-config]
<jms-service default-jms-host="default_JMS_host" type="EMBEDDED"
  <jms-host port="7676" host="localhost" name="default_JMS_host"/>
</jms-service>
Command get-active-module-config executed successfully.
```

### See Also

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

## Administering Resources

This section contains instructions for integrating resources into the Eclipse GlassFish environment. Information about administering specific resources, such as JDBC, is contained in other chapters.

### To Add Resources From an XML File

Use the `add-resources` subcommand in remote mode to create the resources named in the specified XML file. The following resources are supported: JDBC connection pool and resource, JMS, JNDI, and JavaMail resources, custom resource, connector resource and work security map, admin object, and resource adapter configuration.

The XML file must reside in the `domain-dir/config` directory. If you specify a relative path or simply provide the name of the XML file, this subcommand will prepend `domain-dir/config` to this operand.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Add resources from an XML file by using the `add-resources` subcommand.

Information about properties for the subcommand is included in this help page.

### 3. Restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 2-21 Adding Resources

This example creates resources using the contents of the `resource.xml` file on `localhost`.

```
asadmin> add-resources c:\tmp\resource.xml
Command : JDBC resource jdbc1 created successfully.
Command : JDBC connection pool poolA created successfully.
Command add-resources executed successfully.
```

#### See Also

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

## Listing Various System Elements

The following topics are addressed here:

- [To Display the Eclipse GlassFish Version](#)
- [To List Applications](#)
- [To List Containers](#)
- [To List Modules](#)
- [To List Subcommands](#)
- [To List Timers](#)
- [To Show Component Status](#)

### To Display the Eclipse GlassFish Version

Use the `version` subcommand in remote mode to display information about the Eclipse GlassFish version for a particular server. If the subcommand cannot communicate with the server by using the specified login (user/password) and target (host/port) information, then the local version is displayed along with a warning message.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Display the version by using the `version` subcommand.

## Example 2-22 Displaying Version Information

This example displays the version of Eclipse GlassFish on the local host.

```
asadmin> version
Version = Eclipse GlassFish 7.0.0 (build 19)
Command version executed successfully.
```

### See Also

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

## To List Applications

Use the `list-applications` subcommand in remote mode to list the deployed Java applications. If the `--type` option is not specified, all applications are listed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List applications by using the `list-applications` subcommand.

### Example 2-23 Listing Applications

This example lists the web applications on `localhost`.

```
asadmin> list-applications --type web
hellojsp <web>
Command list-applications executed successfully.
```

### See Also

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

## To List Containers

Use the `list-containers` subcommand in remote mode to list application containers.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List containers by using the `list-containers` subcommand.

### Example 2-24 Listing Containers

This example lists the containers on `localhost`.

```
asadmin> list-containers
List all known application containers
Container : grizzly
Container : ejb
Container : webservices
Container : ear
Container : appclient
Container : connector
Container : jpa
Container : web
Container : security
Container : webbeans
Command list-containers executed successfully.
```

## See Also

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

## To List Modules

Use the `list-modules` subcommand in remote mode to list the modules that are accessible to the Eclipse GlassFish module subsystem. The status of each module is included. Possible statuses include NEW and READY.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List modules by using the `list-modules` subcommand.

### Example 2-25 Listing Modules

This example lists the accessible modules.

```
asadmin> list-modules
```

Information similar to the following is displayed (partial output):

```
List Of Modules
Module : org.glassfish.web.jstl-connector:10.0.0.b28
    properties=(visibility=public,State=READY,Sticky=true)
    Module Characteristics : List of Jars implementing the module
        Jar : file:/C:/Preview/v3_Preview_release/distributions/web/target/glass
              fish/modules/web/jstl-connector.jar
    Module Characteristics : List of imported modules
    Module Characteristics : Provides to following services
Module : org.glassfish.admingui.console-common:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
```

```
Module : org.glassfish.admin.launcher:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
Module : org.glassfish.external.commons-codec-repackaged:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
Module : com.sun.enterprise.tiger-types-osgi:0.3.32.Preview-b28
    properties=(visibility=public,State=READY,Sticky=true)
Module Characteristics : List of imported modules
Module Characteristics : Provides to following services
Module Characteristics : List of Jars implementing the module
    Jar : file:/C:/Preview/v3_Preview_release/distributions/web/target/glass
fish/modules/tiger-types-osgi.jar.
...
Command list-modules executed successfully.
```

## See Also

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

## To List Subcommands

Use the `list-commands` subcommand in remote mode to list the deployed `asadmin` subcommands. You can specify that only remote subcommands or only local subcommands are listed. By default, this subcommand displays a list of local subcommands followed by a list of remote subcommands.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List subcommands by using the `list-commands` subcommand.

### Example 2-26 Listing Subcommands

This example lists only local subcommands.

```
asadmin> list-commands --localonly
create-domain
delete-domain
list-commands
list-domains
login
monitor
start-database
start-domain
stop-domain
stop-database
version
Command list-commands executed successfully.
```

## See Also

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

## To List Timers

The timer service is a persistent and transactional notification service that is provided by the enterprise bean container and is used to schedule notifications or events used by enterprise beans. All enterprise beans except stateful session beans can receive notifications from the timer service. Persistent timers set by the service are not destroyed when the server is shut down or restarted.

Use the `list-timers` subcommand in remote mode to list the persistent timers owned by a specific server instance. You can use this information to decide whether to do a timer migration, or to verify that a migration has been completed successfully.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List timers by using the `list-timers` subcommand.

### Example 2-27 Listing Timers

This example lists the timers in a particular standalone server instance. There is one currently active timer set.

```
asadmin> list-timers server  
1  
The list-timers command was executed successfully.
```

## To Show Component Status

Use the `show-component-status` subcommand in remote mode to get the status (either enabled or disabled) of the specified deployed component.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Show component status by using the `show-component-status` subcommand.

### Example 2-28 Showing Status of a Component

This example shows the status of the `MEjbApp` component.

```
asadmin> show-component-status MEjbApp  
Status of MEjbApp is enabled  
Command show-component-status executed successfully.
```

# Using REST Interfaces to Administer Eclipse GlassFish

Eclipse GlassFish provides representational state transfer (REST) interfaces to enable you to access monitoring and configuration data for Eclipse GlassFish, including data that is provided by newly installed add-on components.

You can access the Eclipse GlassFish REST interfaces through client applications such as:

- Web browsers
- [cURL](#)
- [GNU Wget](#)

You can also use the Eclipse GlassFish REST interfaces in REST client applications that are developed in languages such as:

- JavaScript
- Ruby
- Perl
- Java
- JavaFX

The implementation of the Eclipse GlassFish REST interfaces is based on [project Jersey](#). Project Jersey is the reference implementation of [Java Specification Request \(JSR\) 311: JAX-RS: The Java API for RESTful Web Services](#). Information about JSR 311 is also available from the [JSR 311 project home page](#). Information about Jakarta RESTful Web Services is here: <https://jakarta.ee/specifications/restful-ws/>

The following topics are addressed here:

- [Using REST URLs to Administer Eclipse GlassFish](#)
- [Using REST Resource Methods to Administer Eclipse GlassFish](#)
- [Resources for `asadmin` Subcommands That Perform Non-CRUD Operations](#)
- [Securing Eclipse GlassFish REST Interfaces](#)
- [Formats for Resource Representation of Configuration Objects](#)
- [Formats for Resource Representation of Monitoring Objects](#)
- [Formats for Resource Representation of Log File Details](#)
- [Supported Content Types in Requests to REST Resources](#)

## Using REST URLs to Administer Eclipse GlassFish

Each object in the configuration and monitoring object trees is represented as a REST resource that is accessible through an HTTP uniform resource locator (URL). Access to REST resources for Eclipse

GlassFish monitoring and configuration data requires a running DAS.

## REST URLs to Resources for Configuration and Monitoring Objects

The formats of the URLs to resources that represent objects in the configuration and monitoring object trees are as follows:

- Configuration: `http://`host`:`port`/management/domain/`path`
- Monitoring: `http://`host`:`port`/monitoring/domain/`path`

The replaceable items in these URLs are as follows:

### host

The host where the DAS is running.

### port

The HTTP port or HTTPS port for administration.

### path

The path to the object. The path is the dotted name of the object in which each dot (.) is replaced with a slash (/).



The path to a Eclipse GlassFish instance is `servers/server/instance-name`, where instance-name is the name of the instance. For the DAS, instance-name is `server` and the path is `servers/server/server`.

For more information, see the following documentation:

- The [dotted-names\(5ASC\)](#) help page
- [How the Monitoring Tree Structure Works](#)
- [How Dotted Names Work for Configuration](#)

If the URL to a REST resource for Eclipse GlassFish monitoring or configuration data is opened in a web browser, the browser displays a web page that contains the following information about the resource:

- A list of the attributes of the resource and their values. If the resource represents an object in the configuration tree, these attributes are presented in an HTML form that you can use to update the resource. Attributes of a resource for an object in the monitoring tree are read only.
- A list of hypertext links to the children of the resource. This list of links enables you to traverse the tree that contains the resource and to discover the all resources in the tree.
- A list of hypertext links to resources that represent `asadmin` subcommands for non-CRUD operations on the resource.

The following figure shows the web page for the REST resource for managing a domain.

Figure 2-1 Web Page for the REST Resource for Managing a Domain

The screenshot shows a web browser window with the URL <http://localhost:4848/management/domain/nodes/node/sj01>. The page title is "GlassFish REST Interface".

**Node Attributes**

installDir:	/export/glassfish3
nodeDir:	
nodeHost:	sj01.example.com
type:	SSH

**Child Resources**

[application-ref](#)  
[resource-ref](#)  
[ssh-connector](#)

**Commands**

[ping-node-ssh](#)  
[update-node-ssh](#)  
[update-node-config](#)

## REST URLs for Accessing the Log File

The `server.log` file of the DAS is represented as a child that is named `view-log` of the resource for managing the domain. A child of the resource for the `server.log` file represents the log file details

The formats of the URLs to resources that represent the log file are as follows:

- Log file: `http://host:port/management/domain/view-log`
- Log file details: `http://host:port/monitoring/domain/view-log/details`

The replaceable items in these URLs are as follows:

### host

The host where the DAS is running.

### port

The HTTP port or HTTPS port for administration.

You can use the optional `start` parameter in the URL to the resource for the log file to specify the number of characters at the start of the file to skip. For example, to skip 10,000 characters, specify the URL as <http://localhost:4848/management/domain/view-log?start=10000>. This example assumes that the DAS is running on the local host and uses the default port for administration.

The resource for the log file returns the HTTP header "`X-Text-Append-Next`", which contains the entire URL to pass to the `GET` method to return the changes since the last call. You can use this header in client applications to get all log entries that were added in particular interval. For example, by testing the value of the "`X-Text-Append-Next`" header in a client thread every 10 seconds, you can monitor the log entries that were added in the last 10 seconds.

## Using REST Resource Methods to Administer Eclipse GlassFish

The Eclipse GlassFish REST interfaces support methods for accessing objects in the monitoring and configuration object trees.

The following table shows the REST methods for administering monitoring and configuration data and the tasks that you can perform with each method. These methods are HTTP 1.1 primitives. For the detailed specification of these primitives, see [Hypertext Transfer Protocol—HTTP/1.1](#).

Table 2-1 REST Resource Methods for Administering Monitoring and Configuration Data

Task	REST Method
Determine the methods and method parameters that an object in the tree supports	<code>GET</code>
Retrieve data for an object in the tree	<code>GET</code>
Add an object to the tree	<code>POST</code>
Update an object in the tree	<code>POST</code>
Delete an object from the tree	<code>DELETE</code>

REST requests that add, update, or delete objects must specify the `X-Requested-By` header with the value `GlassFish REST HTML interface`.



The `GET` method determines the methods and method parameters that an object in the tree supports and provides additional information about the object. For details, see [To Retrieve Data for an Object in the Tree](#).

### To Determine the Methods and Method Parameters That an Object in the Tree Supports

The methods and method parameters that an object in the tree supports depend on the REST resource that represents the object:

- REST resources for monitoring support only the **GET** method.
- All REST resources for configuration support the **GET** method. However, only some REST resources for configuration also support the **POST** method and the **DELETE** method.

Before performing any operations on an object in the tree, determine the methods and method parameters that the object supports.

You can specify the format in which this information is presented. For more information, see [Formats for Resource Representation of Configuration Objects](#).

 Each **POST** method and **DELETE** method that a REST resource supports has an equivalent **asadmin** subcommand. The parameters of a **POST** method or a **DELETE** method correspond to the options of the method's equivalent **asadmin** subcommand. For information about the options of **asadmin** subcommand, see the [oEclipse GlassFish Reference Manual](#).

1. Ensure that the server is running.

Operations on REST resources for Eclipse GlassFish data require a running server.

2. Use the **GET** method on the REST resource that represents the object.

The **GET** method returns the list of methods that the resource supports. For each method, the list of acceptable message parameters or the list of acceptable query parameters are returned.

Example 2-29 Determining the Methods and Method Parameters That an Object in the Tree Supports

This example uses the cURL utility to determine the methods and method parameters that the resource for the node **sj01** supports. The example uses the following options of the cURL utility:

- **-X** to specify that the **GET** method is used
- **-H** to specify that the resource is represented in JavaScript Object Notation (JSON)

In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The resource supports the **GET** method and the **POST** method.

Line breaks and white space are added to enhance readability.

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/nodes/node/sj01
{
  "command": "Node",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [
      {"path": "_delete-node", "command": "delete-node", "method": "DELETE"},
      {"path": "_update-node", "command": "_update-node", "method": "POST"},
      {"path": "ping-node-ssh", "command": "ping-node-ssh", "method": "GET"},
```

```

    {"path":"update-node-ssh","command":"update-node-ssh","method":"POST"},  

    {"path":"update-node-config","command":"update-node-config","method":"POST"}],  

    "methods": [  

        {"name": "GET"},  

        {"name": "POST", "messageParameters": {  

            "installDir": {"optional": "true", "type": "string", "key": "false"},  

            "nodeDir": {"optional": "true", "type": "string", "key": "false"},  

            "nodeHost": {"optional": "true", "type": "string", "key": "false"},  

            "type": {"optional": "true", "type": "string", "key": "false"}  

        }}  

    ],  

    "entity": {  

        "installDir": "\export\glassfish7",  

        "name": "sj01",  

        "nodeDir": null,  

        "nodeHost":  

        "sj01.example.com",  

        "type": "SSH"  

    },  

    "childResources": {  

        "application-ref":  

        "https://localhost:4848/management/domain/nodes/node/sj01/application-  

        ref",  

        "resource-ref":  

        "https://localhost:4848/management/domain/nodes/node/sj01/resource-  

        ref",  

        "ssh-connector":  

        "https://localhost:4848/management/domain/nodes/node/sj01/ssh-  

        connector"  

    }  

}

```

## To Retrieve Data for an Object in the Tree

Retrieving data for an object in the tree obtains the following information about the REST resource that represents the object:

- A list of the REST methods that the resource supports
- A list of the attributes of the resource and their values
- A list of URLs to the children of the resource

You can specify the format in which this information is presented. For more information, see [Formats for Resource Representation of Configuration Objects](#).

1. Ensure that the server is running.

Operations on REST resources for Eclipse GlassFish data require a running server.

2. Use the **GET** method on the REST resource that represents the object.

#### Example 2-30 Retrieving Data for an Object in the Tree

This example uses the cURL utility to retrieve data for the resource for a the node **sj01**. The example uses the following options of the cURL utility:

- **-X** to specify that the **GET** method is used
- **-H** to specify that the resource is represented in JavaScript Object Notation (JSON)

In this example, the DAS is running on the local host and the HTTP port for administration is 4848.

Line breaks and white space are added to enhance readability.

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/nodes/node/sj01
{
  "command": "Node",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [
      {"path": "_delete-node", "command": "delete-node", "method": "DELETE"},
      {"path": "_update-node", "command": "_update-node", "method": "POST"},
      {"path": "ping-node-ssh", "command": "ping-node-ssh", "method": "GET"},
      {"path": "update-node-ssh", "command": "update-node-ssh", "method": "POST"},
      {"path": "update-node-config", "command": "update-node-config", "method": "POST"}],
    "methods": [
      {"name": "GET"},
      {"name": "POST", "messageParameters": {
        "installDir": {"optional": "true", "type": "string", "key": "false"}, "nodeDir": {"optional": "true", "type": "string", "key": "false"}, "nodeHost": {"optional": "true", "type": "string", "key": "false"}, "type": {"optional": "true", "type": "string", "key": "false"}}
      }
    ],
    "entity": {
      "installDir": "\\\export\\glassfish7",
      "name": "sj01",
      "nodeDir": null,
      "nodeHost": "sj01.example.com",
      "type": "SSH"
    },
    "childResources": {
      "application-ref": "https:\\\\localhost:4848\\management\\domain\\nodes\\node\\sj01\\application-ref",
      "resource-ref": ...
    }
  }
}
```

```
"https://localhost:4848/management/domain/nodes/node/sj01/resource-ref",
    "ssh-connector":
        "https://localhost:4848/management/domain/nodes/node/sj01/ssh-connector"
    }
}
}
```

## To Add an Object to the Tree

1. Ensure that the server is running.

Operations on REST resources for Eclipse GlassFish data require a running server.

2. Determine the acceptable message parameters for the **POST** method of the resource that represents the parent of the object.

For information about how to perform this step, see [To Determine the Methods and Method Parameters That an Object in the Tree Supports](#).

3. Use the **POST** method on the REST resource that represents the parent of the object that you are adding.
4. Confirm that the object has been added.

Perform this step on the resource that represents the object that you have just added, not the parent. For information about how to perform this step, see [To Retrieve Data for an Object in the Tree](#).

### Example 2-31 Adding an Object to the Tree

This example uses the cURL utility to add a JDBC resource object to the tree by creating a REST resource to represent the JDBC resource.

In this example, the DAS is running on the local host and the HTTP port for administration is 4848.

Line breaks are added to enhance readability.

1. This step determines the acceptable message parameters for the **POST** method of the resource **jdbc-resource**.

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/jdbc-resource
{
    "command":"Jdbc-resource",
    "exit_code":"SUCCESS",
    "extraProperties":{
        "commands":[] ,
```

```

"methods": [
    {"name": "GET"},
    {"name": "POST", "messageParameters": {

"description": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}, 

"enabled": {"acceptableValues": "", "optional": "true", "type": "boolean", "defaultValue": "true"}, 

"id": {"acceptableValues": "", "optional": "false", "type": "string", "defaultValue": ""}, 

"poolName": {"acceptableValues": "", "optional": "false", "type": "string", "defaultValue": ""}, 

"property": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}, 

"target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""} 
    } 
},
],
"childResources": {
    "jdbc\_\_TimerPool": 
        "https://localhost:4848/management/domain/resources/jdbc-resource/jdbc\_\_TimerPool",
    "jdbc\_\_default": 
        "https://localhost:4848/management/domain/resources/jdbc-resource/jdbc\_\_default"
}
}
}

```

2. This step adds a resource as a child of the `jdbc-resource` resource. The `-d` option of the cURL utility sets the required message parameters as follows:

- `id` is set to `jdbc/myjdbcresource`.
- `connectionpoolid` is set to `DerbyPool`.

```

curl -X POST -H "X-Requested-By: GlassFish REST HTML interface"
-d id=jdbc/myjdbcresource -d connectionpoolid=DerbyPool
http://localhost:4848/management/domain/resources/jdbc-resource

```

3. This step confirms that the object has been added by retrieving data for the REST resource that represents the object.

```

curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/

```

```

jdbc-resource/jdbc%2Fmyjdbcresource
{
  "command": "Jdbc-resource",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [],
    "methods": [
      {"name": "GET"},
      {"name": "POST", "messageParameters": {
        "description": {"optional": "true", "type": "string", "key": "false"},

        "enabled": {"optional": "true", "type": "boolean", "defaultValue": "true", "key": "false"},

        "jndiName": {"optional": "true", "type": "string", "key": "true"},

        "objectType": {"optional": "true", "type": "string", "defaultValue": "user", "key": "false"},

        "poolName": {"optional": "true", "type": "string", "key": "false"}
      }},
      {"name": "DELETE", "messageParameters": {

        "target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}
      }}
    ],
    "childResources": {
      "property": {
        "https://localhost:4848/management/domain/resources/jdbc-
resource/jdbc%2Fmyjdbcresource/property"
      }
    }
  }
}

```

## To Update an Object in the Tree

1. Ensure that the server is running. Operations on REST resources for Eclipse GlassFish data require a running server.
2. Determine the acceptable message parameters for the **POST** method of the resource that represents the object. For information about how to perform this step, see [To Determine the Methods and Method Parameters That an Object in the Tree Supports](#).
3. Use the **POST** method on the REST resource that represents the object that you are updating.
4. Confirm that the object has been updated. For information about how to perform this step, see [To Retrieve Data for an Object in the Tree](#).

Example 2-32 Updating an Object in the Tree

This example uses the cURL utility to update a JDBC resource in the tree by modifying the REST resource that represents the JDBC resource.

In this example, the DAS is running on the local host and the HTTP port for administration is 4848.

Line breaks are added to enhance readability.

1. This step determines the acceptable message parameters for the **POST** method of the resource **jdbc-myjdbcresource**.

```
curl -X OPTIONS -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
{
  "command": "Jdbc-resource",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [],
    "methods": [
      {"name": "GET"},
      {"name": "POST", "messageParameters": {
        "description": {"optional": "true", "type": "string", "key": "false"},

        "enabled": {"optional": "true", "type": "boolean", "defaultValue": "true", "key": "false"},

        "jndiName": {"optional": "true", "type": "string", "key": "true"},

        "objectType": {"optional": "true", "type": "string", "defaultValue": "user", "key": "false"},

        "poolName": {"optional": "true", "type": "string", "key": "false"}
      }}
    ],
    {"name": "DELETE", "messageParameters": {

      "target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}
    }
  }
},
  "childResources": {
    "property": {
      "https://localhost:4848/management/domain/resources/jdbc-
resource/jdbc%2Fmyjdbcresource/property"
    }
  }
}
```

2. This step updates the REST resource **jdbc-myjdbcresource** to disable the JDBC resource that **jdbc-myjdbcresource** represents. The **-d** option of the cURL utility sets the **enabled** message parameter to **disabled**.

```
curl -X POST -H "X-Requested-By: GlassFish REST HTML interface"  
-d "enabled=false" http://localhost:4848/management/domain/resources/  
jdbc-resource/jdbc%2Fmyjdbcresource
```

3. This step confirms that the object has been updated by retrieving data for the REST resource that represents the object.

```
curl -X GET -H "Accept: application/json"  
http://localhost:4848/management/domain/resources/  
jdbc-resource/jdbc%2Fmyjdbcresource  
{  
    "command": "Jdbc-resource",  
    "exit_code": "SUCCESS",  
    "extraProperties": {  
        "commands": [],  
        "methods": [  
            {"name": "GET"},  
            {"name": "POST", "messageParameters": {  
                "description": {"optional": "true", "type": "string", "key": "false"},  
  
                "enabled": {"optional": "true", "type": "boolean", "defaultValue": "true", "key": "false"},  
                "jndiName": {"optional": "true", "type": "string", "key": "true"},  
                "objectType": {"optional": "true", "type": "string", "defaultValue":  
                    "user", "key": "false"},  
                "poolName": {"optional": "true", "type": "string", "key": "false"}  
            }  
        ],  
        {"name": "DELETE", "messageParameters": {  
  
            "target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}  
        }  
    }  
},  
    "entity": {  
        "description": null,  
        "enabled": "false",  
        "jndiName": "jdbc\myjdbcresource",  
        "objectType":  
            "user",  
        "poolName": "DerbyPool"  
    },  
    "childResources": {  
        "property":  
            "https://localhost:4848/management/domain/resources/jdbc-resource/  
                jdbc%2Fmyjdbcresource/property"  
    }  
}
```

```
}
```

## To Delete an Object From the Tree

1. Ensure that the server is running.

Operations on REST resources for Eclipse GlassFish data require a running server.

2. Confirm that the object can be deleted.

For information about how to perform this step, see [To Determine the Methods and Method Parameters That an Object in the Tree Supports](#).

3. Confirm that the object has been deleted.

Perform this step on the resource that represents the parent of the object that you have just deleted. For information about how to perform this step, see [To Retrieve Data for an Object in the Tree](#).

### Example 2-33 Deleting an Object From the Tree

This example uses the curl utility to delete a JDBC resource from the tree by deleting the REST resource that represents the JDBC resource.

In this example, the DAS is running on the local host and the HTTP port for administration is 4848.

Line breaks and white space are added to enhance readability.

1. This step confirms that the object can be deleted by retrieving the REST methods that the resource `jdbc-myjdbcresource` supports.

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc%2Fmyjdbcresource
{
  "command":"Jdbc-resource",
  "exit_code":"SUCCESS",
  "extraProperties":{
    "commands":[],
    "methods":[
      {"name":"GET"},
      {"name":"POST","messageParameters":{
        "description":{"optional":"true","type":"string","key":"false"},

      "enabled":{"optional":"true","type":"boolean","defaultValue":"true","key":"false"},

      "jndiName":{"optional":"true","type":"string","key":"true"},

      "objectType":{"optional":"true","type":"string","defaultValue":"user","key":"false"}},

    ]}}}
```

```

    "poolName": {"optional": "true", "type": "string", "key": "false"}
  }
},
{"name": "DELETE", "messageParameters": {

  "target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}
}
  }
],
"childResources": {
  "property": {
    "https://localhost:4848/management/domain/resources/jdbc-resource/
      jdbc%2Fmyjdbcresource/property"
  }
}
}
}

```

- This step deletes the **jdbc/myjdbcresource** resource.

```

curl -X DELETE -H "X-Requested-By: GlassFish REST HTML interface"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc%2Fmyjdbcresource

```

- This step confirms that the object has been deleted by retrieving data for the REST resource that represents the parent of the object.

```

curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/jdbc-resource
{
  "command": "Jdbc-resource",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [],
    "methods": [
      {"name": "GET"},
      {"name": "POST", "messageParameters": {

        "description": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""},

        "enabled": {"acceptableValues": "", "optional": "true", "type": "boolean", "defaultValue": "true"},

        "id": {"acceptableValues": "", "optional": "false", "type": "string", "defaultValue": ""},

        "poolName": {"acceptableValues": "", "optional": "false", "type": "string", "defaultValue": ""}
      }, ...
    ]
  }
}

```

```

"property": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""},  

"target": {"acceptableValues": "", "optional": "true", "type": "string", "defaultValue": ""}  

    }  

}  

]  

"childResources": {  

    "jdbc\_\_TimerPool":  

        "https://localhost:4848/management/domain/resources/jdbc-resource/jdbc%2F__TimerPool",  

    "jdbc\_\_default":  

        "https://localhost:4848/management/domain/resources/jdbc-resource/jdbc%2F__default"  

    }  

}
}

```

## Resources for `asadmin` Subcommands That Perform Non-CRUD Operations

The Eclipse GlassFish REST interfaces also support operations other than create, read, update, and delete (CRUD) operations, for example:

- State management
- Queries
- Application deployment

These operations are supported through REST resources that represent the `asadmin` subcommands for performing these operations. Each resource is a child of the resource on which the operation is performed. The child resources do not represent objects in the configuration object tree.

For example, the resource that represents a node provides child resources for the following `asadmin` subcommands that perform non-CRUD operations on the node:

- `ping-node-ssh`
- `update-node-config`
- `update-node-ssh`

## Securing Eclipse GlassFish REST Interfaces

The Eclipse GlassFish REST interfaces support the following authentication schemes for securing the REST interfaces:

- Basic authentication over a secure connection
- Authentication by using session tokens

When security is enabled, you must specify [https](#) as the protocol in the URLs to REST resources and provide a user name and password.

## Setting Up Basic Authentication Over a Secure Connection

Setting up basic authentication over a secure connection to secure Eclipse GlassFish REST interfaces involves the following sequence of tasks:

1. Adding an [admin-realm](#) user to the [asadmin](#) user group
2. Enabling Secure Sockets Layer (SSL)

For information about how to perform these tasks from the command line, see the following documentation:

- "[To Create an Authentication Realm](#)" in Eclipse GlassFish Security Guide
- "[To Create a File User](#)" in Eclipse GlassFish Security Guide
- [To Configure an HTTP Listener for SSL](#)

For information about how to perform these tasks by using the Administration Console, see the following topics in the Administration Console online help:

- To Add a User to the Admin Realm
- To Edit SSL Settings for a Protocol

## To Secure REST Interfaces by Using Session Tokens

Basic authentication requires a REST client to cache a user's credentials to enable the client to pass the credentials with each request. If you require a REST client not to cache credentials, your client must use session tokens for authentication.

1. Request a session token by using the [GET](#) method on the resource at <http://host:port/management/sessions>. Eclipse GlassFish uses basic authentication to authenticate the client, generates a session token, and passes the token to the client.
2. In each subsequent request that requires authentication, use the token to authenticate the client.
  - a. Create a cookie that is named [gfrestrtoken](#) the value of which is the token.
  - b. Send the cookie with the request.
  - c. When the token is no longer required, retire the token by using the [DELETE](#) method on the resource at <http://host:port/management/sessions/{tokenvalue}>.



If a client does not explicitly retire a token, the token is retired after 30 minutes of inactivity.

## Formats for Resource Representation of Configuration Objects

The Eclipse GlassFish REST interfaces represent resources for configuration objects in the following formats:

- [JSON](#)
- [XML](#)
- [HTML](#)

Eclipse GlassFish enables you to specify the resource representation through the filename extension in the URL or through the HTTP header:

- To specify the resource representation through the filename extension in the URL, specify the appropriate extension as follows:
  - For JSON, specify the `.json` extension.
  - For XML, specify the `.xml` extension.
  - For HTML, omit the extension.
- How to specify the resource representation through the HTTP header depends on the client that you are using to access the resource. For example, if you are using the cURL utility, specify the resource representation through the `-H` option as follows:
  - For JSON, specify `-H "Accept: application/json"`.
  - For XML, specify `-H "Accept: application/xml"`.
  - For HTML, omit the `-H` option.

## JSON Resource Representation for Configuration Objects

The general format for the JSON representation of a resource for a configuration object is as follows:

```
{  
  "command": "resource",  
  "exit_code": "code",  
  "extraProperties": {  
    "commands": [command-list],  
    "methods": [method-list],  
    "entity": {attributes},  
    "childResources": {children}  
  }  
}
```

```
}
```

The replaceable items in this format are as follows:

#### **resource**

The name of the resource.

#### **code**

The result of the attempt to get the resource.

#### **command-list**

One or more metadata sets separated by a comma (,) that represent the `asadmin` subcommands for performing non—CRUD operations on the resource. For the format of each metadata set, see [JSON Representation of a Command in a Command List](#).

#### **method-list**

One or more metadata sets separated by a comma (,) that represent the methods that the resource supports. For the format of each metadata set, see [JSON Representation of a Method in a Method List](#).

#### **attributes**

Zero or more name-value pairs separated by a comma (,). Each name-value pair is specified as `'name': 'value'`.

#### **children**

Zero or more child resources separated by a comma (,). Each child resource is specified as `"resource-name": "url"`.

#### **resource-name**

The name of the resource as displayed in client applications that access the parent of the resource.

#### **url**

The URL to the child resource.

### [JSON Representation of a Command in a Command List](#)

The JSON representation of a command in a command list is as follows:

```
{
  "path": "command-path",
  "command": "command-name",
  "method": "rest-method"
}
```

The replaceable items in this format are as follows:

## **command-path**

The relative path to REST resource that represents the command. This path is relative to the URL of the REST resource that is the parent of the resource that represents the command.

## **command-name**

The name of the command as displayed in client applications that access the resource.

## **rest-method**

The REST resource method that the command invokes when the command is run. The method is **GET**, **POST**, or **DELETE**.

## JSON Representation of a Method in a Method List

The JSON representation of a method in a method list is as follows:

```
{  
  "name": "method-name",  
  "messageParameters": {  
    message-parameter-list  
  },  
  "queryParameters": {  
    queryparameter- list  
  }  
}
```

The replaceable items in this format are as follows:

### **method-name**

The name of the method, which is **GET**, **POST**, or **DELETE**.

### **message-parameter-list**

Zero or more metadata sets separated by a comma (,) that represent the message parameters that are allowed for the method. For the format of each metadata set, see [JSON Representation of a Message Parameter or a Query Parameter](#).

### **query-parameter-list**

Zero or more metadata sets separated by a comma (,) that represent the query parameters that are allowed for the method. For the format of each metadata set, see [JSON Representation of a Message Parameter or a Query Parameter](#).

## JSON Representation of a Message Parameter or a Query Parameter

The JSON representation of a message parameter or a query parameter is as follows:

```
"parameter-name":{attribute-list}
```

The replaceable items in this format are as follows:

### **parameter-name**

The name of the parameter.

### **attribute-list**

A comma-separated list of name-value pairs of attributes for the parameter. Each pair is in the following format:

```
"name": "value"
```

Possible attributes are as follows:

#### **defaultValue**

The default value of the parameter.

#### **acceptableValues**

The set or range of acceptable values for the parameter.

#### **type**

The data type of the parameter, which is one of the following types:

- **boolean**
- **int**
- **string**

#### **optional**

Indicates whether the parameter is optional. If **true**, the parameter is optional. If **false**, the parameter is required.

#### **key**

Indicates whether the parameter is key. If **true**, the parameter is key. If **false**, the parameter is not key.

## Example JSON Resource Representation for a Configuration Object

This example shows the JSON representation of the resource for the node **sj01**. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/management/domain/nodes/node/sj01>.

Line breaks and white space are added to enhance readability.

```
{
  "command": "Node",
  "exit_code": "SUCCESS",
  "extraProperties": {
    "commands": [
      {"path": "_delete-node", "command": "delete-node", "method": "DELETE"},  

      {"path": "_update-node", "command": "_update-node", "method": "POST"},  

      {"path": "ping-node-ssh", "command": "ping-node-ssh", "method": "GET"},  

      {"path": "update-node-ssh", "command": "update-node-ssh", "method": "POST"},  

      {"path": "update-node-config", "command": "update-node-config", "method": "POST"}],
    "methods": [
      {"name": "GET"},  

      {"name": "POST", "messageParameters": {
        "installDir": {"optional": "true", "type": "string", "key": "false"},  

        "nodeDir": {"optional": "true", "type": "string", "key": "false"},  

        "nodeHost": {"optional": "true", "type": "string", "key": "false"},  

        "type": {"optional": "true", "type": "string", "key": "false"}  

      }}
    ]
  },
  "entity": {
    "installDir": "\export\glassfish7",
    "name": "sj01",
    "nodeDir": null,
    "nodeHost":  

    "sj01.example.com",
    "type": "SSH"
  },
  "childResources": {
    "application-ref":  

    "https://localhost:4848/management/domain/nodes/node/sj01/application-ref",
    "resource-ref":  

    "https://localhost:4848/management/domain/nodes/node/sj01/resource-ref",
    "ssh-connector":  

    "https://localhost:4848/management/domain/nodes/node/sj01/ssh-connector"
  }
}
}
```

## XML Resource Representation for Configuration Objects

The general format for the XML representation of a resource for a configuration object is as follows:

```

<map>
  <entry key="extraProperties">
    <map>
      <entry key="methods">
        <list>
          methods
        </list>
      </entry>
      <entry key="entity">
        <map>
          attributes
        </map>
      </entry>
      <entry key="commands">
        <list>
          commands
        </list>
      </entry>
      <entry key="childResources">
        <map>
          children
        </map>
      </entry>
    </map>
  </entry>
  <entry key="message"></entry>
  <entry key="exit_code" value="code"></entry>
  <entry key="command" value="resource"></entry>
</map>

```

The replaceable items in this format are as follows:

### methods

One or more XML elements that represent the methods that the resource supports. For the format of each element, see [XML Representation of a Resource Method](#).

### attributes

Zero or more XML elements that represent the attributes of the resource. Each element specifies a name-value pair as follows:

```
<entry key="name" value="value"></entry>
```

### commands

One or more XML elements that represent the `asadmin` subcommands for performing non—CRUD operations on the resource. For the format of each element, see [XML Representation of a Command](#).

## **children**

Zero or more XML elements that represent the children of the resource. Each element is specified as follows:

```
<entry key="resource-name" value="url"></entry>
```

### **resource-name**

The name of the resource as displayed in client applications that access the parent of the resource.

### **url**

The URL to the child resource.

## **code**

The result of the attempt to get the resource.

## **resource**

The name of the resource.

## XML Representation of a Resource Method

The XML representation of a method in a method list is as follows:

```
<map>
<entry key="name" value="method-name"></entry>
<entry key="messageParameters">
  message-parameter-list
</entry>
<entry key="queryParameters">
  message-parameter-list
</entry>
</map>
```

The replaceable items in this format are as follows:

### **method-name**

The name of the method, which is **GET**, **POST**, or **DELETE**.

### **message-parameter-list**

Zero or more XML elements that represent the message parameters that are allowed for the method. For the format of each element, see [XML Representation of a Message Parameter or a Query Parameter](#).

### **query-parameter-list**

Zero or more XML elements that represent the query parameters that are allowed for the

method. For the format of each element, see [XML Representation of a Message Parameter or a Query Parameter](#).

## XML Representation of a Command

The XML representation of a command is as follows:

```
<map>
<entry key="command" value="command-name"></entry>
<entry key="path" value="command-path"></entry>
<entry key="method" value="rest-method"></entry>
</map>
```

The replaceable items in this format are as follows:

### **command-name**

The name of the command as displayed in client applications that access the resource.

### **command-path**

The relative path to REST resource that represents the command. This path is relative to the URL of the REST resource that is the parent of the resource that represents the command.

### **rest-method**

The REST resource method that the command invokes when the command is run. The method is **GET**, **POST**, or **DELETE**.

## XML Representation of a Message Parameter or a Query Parameter

The XML representation of a message parameter or a query parameter is as follows:

```
<map>
<entry key="parameter-name">
<map>
  attributes
</map>
</entry>
</map>
```

The replaceable items in this format are as follows:

### **parameter-name**

The name of the parameter.

## attributes

One or more XML elements that represent the attributes for the parameter. Each element specifies a name-value pair as follows:

```
<entry key="name" value="value"></entry>
```

Possible attributes are as follows:

### defaultValue

The default value of the parameter.

### acceptablevalues

The set or range of acceptable values for the parameter.

### type

The data type of the parameter, which is one of the following types:

- boolean
- int
- string

### optional

Indicates whether the parameter is optional. If true, the parameter is optional. If false, the parameter is required.

### key

Indicates whether the parameter is key. If true, the parameter is key. If false, the parameter is not key.

## Example XML Resource Representation

This example shows the XML representation of the resource for the node sj01. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/management/domain/nodes/node/sj01>.

Line breaks and white space are added to enhance readability.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<map>
  <entry key="extraProperties">
    <map>
      <entry key="methods">
        <list>
          <map>
            <entry key="name" value="GET"/>
          </map>
        </list>
      </entry>
    </map>
  </entry>
</map>
```

```

<map>
  <entry key="name" value="POST"/>
  <entry key="messageParameters">
    <map>
      <entry key="installDir">
        <map>
          <entry key="optional" value="true"/>
          <entry key="type" value="string"/>
          <entry key="key" value="false"/>
        </map>
      </entry>
      <entry key="nodeDir">
        <map>
          <entry key="optional" value="true"/>
          <entry key="type" value="string"/>
          <entry key="key" value="false"/>
        </map>
      </entry>
      <entry key="type">
        <map>
          <entry key="optional" value="true"/>
          <entry key="type" value="string"/>
          <entry key="key" value="false"/>
        </map>
      </entry>
      <entry key="nodeHost">
        <map>
          <entry key="optional" value="true"/>
          <entry key="type" value="string"/>
          <entry key="key" value="false"/>
        </map>
      </entry>
    </map>
  </entry>
</map>
</list>
</entry>
<entry key="entity">
  <map>
    <entry key="installDir" value="/export/glassfish7"/>
    <entry key="name" value="sj01"/>
    <entry key="nodeDir" value="" />
    <entry key="type" value="SSH"/>
    <entry key="nodeHost" value="sj01example.com"/>
  </map>
</entry>
<entry key="commands">
  <list>
    <map>
      <entry key="command" value="delete-node"/>
      <entry key="path" value="_delete-node"/>
    </map>
  </list>
</entry>

```

```

<entry key="method" value="DELETE"/>
</map>
<map>
<entry key="command" value="_update-node"/>
<entry key="path" value="_update-node"/>
<entry key="method" value="POST"/>
</map>
<map>
<entry key="command" value="ping-node-ssh"/>
<entry key="path" value="ping-node-ssh"/>
<entry key="method" value="GET"/>
</map>
<map>
<entry key="command" value="update-node-ssh"/>
<entry key="path" value="update-node-ssh"/>
<entry key="method" value="POST"/>
</map>
<map>
<entry key="command" value="update-node-config"/>
<entry key="path" value="update-node-config"/>
<entry key="method" value="POST"/>
</map>
</list>
</entry>
<entry key="childResources">
<map>
<entry key="application-ref"
      value="https://localhost:4848/management/domain/nodes/node/sj01/application-ref"
/>
<entry key="ssh-connector"
      value="https://localhost:4848/management/domain/nodes/node/sj01/ssh-connector"/>
<entry key="resource-ref"
      value="https://localhost:4848/management/domain/nodes/node/sj01/resource-ref"/>
</map>
</entry>
</map>
</entry>
<entry key="message"/>
<entry key="exit_code" value="SUCCESS"/>
<entry key="command" value="Node"/>
</map>

```

## HTML Resource Representation for Configuration Objects

The format for the HTML representation of a resource for a configuration object is a web page that provides the following information about the resource:

- A list of the attributes of the resource and their values.

- A list of the methods and method parameters that the resource supports. Each method and its parameters are presented as a field of the appropriate type in an HTML form.
- A list of hypertext links to the children of the resource.
- A list of hypertext links to resources that represent `asadmin` subcommands for non-CRUD operations on the resource.

For a sample web page, see [Figure 2-1](#). In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/management/domain/nodes/node/sj01>.

## Formats for Resource Representation of Monitoring Objects

The Eclipse GlassFish REST interfaces represent resources for monitoring data in the following formats:

- [JSON](#)
- [XML](#)
- [HTML](#)

### JSON Resource Representation for Monitoring Objects

The general format for the JSON representation of a resource for a monitoring object is as follows:

```
{
  "message": "",
  "command": "Monitoring Data",
  "exit_code": "code",
  "extraProperties": {
    "entity": {
      statistics-list
    },
    "childResources": {
      children
    }
  }
}
```

The replaceable items in this format are as follows:

#### **code**

The result of the attempt to get the resource.

## **statistics-list**

Zero or more metadata sets separated by a comma (,) that represent the statistics that the monitoring object provides. For the format of each metadata set, see [JSON Representation of a Statistic in a Statistics List](#).

## **children**

Zero or more child resources separated by a comma (,). Each child resource is specified as "resource-name":"url".

## **resource-name**

The name of the resource as displayed in client applications that access the parent of the resource.

## **url**

The URL to the child resource.

## JSON Representation of a Statistic in a Statistics List

The JSON representation of a counter statistic in a statistics list is as follows:

```
"statistic":{  
    "count":count,  
    "lastsampletime":last-sample-time,  
    "description": "description",  
    "unit": "unit",  
    "name": "name",  
    "starttime":start-time  
}
```

The JSON representation of a range statistic in a statistics list is as follows:

```
"statistic":{  
    "highwatermark":highest-value,  
    "lowwatermark":lowest-value,  
    "current":current-value  
    "lastsampletime":last-sample-time,  
    "description": "description",  
    "unit": "unit",  
    "name": "name",  
    "starttime":start-time  
}
```

The replaceable items in these formats are as follows:

## **statistic**

The name of the statistic.

## **count**

Counter statistics only: The current value of the statistic.

## **highest-value**

Range statistics only: The highest value of the statistic since monitoring of the statistic began.

## **lowest-value**

Range statistics only: The lowest value of the statistic since monitoring of the statistic began.

## **current-value**

Range statistics only: The lowest value of the statistic since monitoring of the statistic began.

## **last-sample-time**

The time in UNIX time at which the statistic was last sampled.

## **description**

A textual description of what the statistic represents.

## **unit**

The unit of measurement of the statistic, which is one of the following units of measurement:

### **count**

The cumulative value of an attribute that increases with time.

### **range**

The lowest value, highest value, and current value of an attribute that can increase or decrease with time.

### **boundedrange**

The lowest value, highest value, and current value of an attribute that can increase or decrease with time and has fixed limits.

### **string**

A string that represents an attribute value. A string statistic is similar to a count, except that the values are not ordered. Typically, a string statistic represents the state of an object, for example, **CONNECTED**, **CLOSED**, or **DISCONNECTED**.

### **time**

Values of an attribute that provide the following timing measurements for an operation:

- The number of times the operation was performed.
- The maximum amount of time to perform the operation once.
- The minimum amount of time to perform the operation once.
- The total amount of time that has been spent performing the operation.
- The average amount of time to perform the operation.

## **name**

The name of the statistic as displayed in client applications that access the resource that contains the statistic.

## **start-time**

The time in UNIX time at which monitoring of the statistic began.

## Example JSON Resource Representation for a Monitoring Object

This example shows the JSON representation of the monitoring object that provides class loader statistics for the virtual machine for the Java platform. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/monitoring/domain/server/jvm/class-loading-system>.

Line breaks and white space are added to enhance readability.

```
{  
  "message": "",  
  "command": "Monitoring Data",  
  "exit_code": "SUCCESS",  
  "extraProperties": {  
    "entity": {  
      "loadedclass-count": {  
        "count": 8521,  
        "lastsampletime": 1300726961018,  
        "description": "Number of classes currently loaded in the Java virtual  
          machine",  
        "unit": "count",  
        "name": "LoadedClassCount",  
        "starttime": 1300483924126  
      },  
      "totalloadedclass-count": {  
        "count": 8682,  
        "lastsampletime": 1300726961018,  
        "description": "Total number of classes that have been loaded since the  
          Java virtual machine has started execution",  
        "unit": "count",  
        "name": "TotalLoadedClassCount",  
        "starttime": 1300483924127  
      },  
      "unloadedclass-count": {  
        "count": 161,  
        "lastsampletime": 1300726961018,  
        "description": "Total number of classes unloaded since the Java virtual  
          machine has started execution",  
        "unit": "count",  
        "name": "UnLoadedClassCount",  
        "starttime": 1300483924127  
      }  
    }  
  }  
}
```

```
        }
    }, "childResources": {}
}
}
```

## XML Resource Representation for Monitoring Objects

The general format for the XML representation of a resource for a monitoring object is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<map>
<entry key="extraProperties">
<map>
<entry key="entity">
<map>
  statistics
</map>
</entry>
<entry key="childResources">
<map>
  children
</map>
</entry>
</map>
</entry>
<entry key="message" value=""></entry>
<entry key="exit_code" value="code"></entry>
<entry key="command" value="Monitoring Data"></entry>
</map>
```

The replaceable items in this format are as follows:

### **statistics**

Zero or more XML elements that represent the statistics that the monitoring object provides. For the format of each element, see [XML Representation of a Statistic](#).

### **children**

Zero or more XML elements that represent the children of the resource. Each element is specified as follows:

```
<entry key="resource-name" value="url"></entry>
```

### **resource-name**

The name of the resource as displayed in client applications that access the parent of the resource.

## url

The URL to the child resource.

## code

The result of the attempt to get the resource.

## XML Representation of a Statistic

The XML representation of a counter statistic is as follows:

```
<entry key="statistic">
  <map>
    <entry key="unit" value="unit"></entry>
    <entry key="starttime">
      <number>start-time</number>
    </entry>
    <entry key="count">
      <number>count</number>
    </entry>
    <entry key="description" value="description"></entry>
    <entry key="name" value="name"></entry>
    <entry key="lastsampletime">
      <number>last-sample-time</number>
    </entry>
  </map>
</entry>
```

The XML representation of a range statistic is as follows:

```
<entry key="statistic">
  <map>
    <entry key="unit" value="unit"></entry>
    <entry key="starttime">
      <number>start-time</number>
    </entry>
    <entry key="highwatermark">
      <number>highest-value</number>
    </entry>
    <entry key="lowwatermark">
      <number>lowest-value</number>
    </entry>
    <entry key="current">
      <number>current-value</number>
    </entry>
    <entry key="description" value="description"></entry>
    <entry key="name" value="name"></entry>
    <entry key="lastsampletime">
```

```
<number>last-sample-time</number>
</entry>
</map>
</entry>
```

The replaceable items in these formats are as follows:

### **statistic**

The name of the statistic.

### **unit**

The unit of measurement of the statistic, which is one of the following units of measurement:

#### **count**

The cumulative value of an attribute that increases with time.

#### **range**

The lowest value, highest value, and current value of an attribute that can increase or decrease with time.

#### **boundedrange**

The lowest value, highest value, and current value of an attribute that can increase or decrease with time and has fixed limits.

#### **string**

A string that represents an attribute value. A string statistic is similar to a count, except that the values are not ordered. Typically, a string statistic represents the state of an object, for example, **CONNECTED**, **CLOSED**, or **DISCONNECTED**.

#### **time**

Values of an attribute that provide the following timing measurements for an operation:

- The number of times the operation was performed.
- The maximum amount of time to perform the operation once.
- The minimum amount of time to perform the operation once.
- The total amount of time that has been spent performing the operation.
- The average amount of time to perform the operation.

### **start-time**

The in time in UNIX time at which monitoring of the statistic began.

### **count**

Counter statistics only: The current value of the statistic.

### **highest-value**

Range statistics only: The highest value of the statistic since monitoring of the statistic began.

## **lowest-value**

Range statistics only: The lowest value of the statistic since monitoring of the statistic began.

## **current-value**

Range statistics only: The lowest value of the statistic since monitoring of the statistic began.

## **description**

A textual description of what the statistic represents.

## **name**

The name of the statistic as displayed in client applications that access the resource that contains the statistic.

## **last-sample-time**

The time in UNIX time at which the statistic was last sampled.

## Example XML Resource Representation for a Monitoring Object

This example shows the XML representation of the monitoring object that provides class loader statistics for the virtual machine for the Java platform. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/monitoring/domain/server/jvm/class-loading-system>.

Line breaks and white space are added to enhance readability.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<map>
  <entry key="extraProperties">
    <map>
      <entry key="entity">
        <map>
          <entry key="unloadedclass-count">
            <map>
              <entry key="unit" value="count"/>
              <entry key="starttime">
                <number>1300483924127</number>
              </entry>
              <entry key="count">
                <number>161</number>
              </entry>
              <entry key="description" value="Total number of classes unloaded since
                the Java virtual machine has started execution"/>
              <entry key="name" value="UnLoadedClassCount"/>
              <entry key="lastsampletime">
                <number>1300726989505</number>
              </entry>
            </map>
          </entry>
        </map>
      </entry>
    </map>
  </entry>
<entry key="totalloadedclass-count">
```

```

<map>
    <entry key="unit" value="count"/>
    <entry key="starttime">
        <number>1300483924127</number>
    </entry>
    <entry key="count">
        <number>8682</number>
    </entry>
    <entry key="description" value="Total number of classes that have been
        loaded since the Java virtual machine has started execution"/>
    <entry key="name" value="TotalLoadedClassCount"/>
    <entry key="lastsampletime">
        <number>1300726989505</number>
    </entry>
</map>
</entry>
<entry key="loadedclass-count">
<map>
    <entry key="unit" value="count"/>
    <entry key="starttime">
        <number>1300483924126</number>
    </entry><entry key="count">
        <number>8521</number>
    </entry>
    <entry key="description" value="Number of classes currently loaded in
        the Java virtual machine"/>
    <entry key="name" value="LoadedClassCount"/>
    <entry key="lastsampletime">
        <number>1300726989505</number>
    </entry>
</map>
</entry>
</map>
</entry>
<entry key="childResources">
    <map/>
</entry>
</map>
</entry>
<entry key="message" value="" />
<entry key="exit_code" value="SUCCESS" />
<entry key="command" value="Monitoring Data" />
</map>

```

## HTML Resource Representation for Monitoring Objects

The format for the HTML representation of a resource for a monitoring object is a web page that provides the following information about the resource:

- A list of the statistics that the resource provides.
- A list of hypertext links to the children of the resource.

The following figure shows the web page for the REST resource that provides class loader statistics for the virtual machine for the Java platform.

**Figure 2-2 Web Page for the REST Resource That Provides Class Loader Statistics**



## GlassFish REST Interface

- [loadedclass-count](#)
  - count : 9787
  - lastsampletime : 1300741760533
  - description : Number of classes currently loaded in the Java virtual machine
  - unit : count
  - name : LoadedClassCount
  - starttime : 1300483924126
- [totalloadedclass-count](#)
  - count : 9972
  - lastsampletime : 1300741760533
  - description : Total number of classes that have been loaded since the Java virtual machine has started execution
  - unit : count
  - name : TotalLoadedClassCount
  - starttime : 1300483924127
- [unloadedclass-count](#)
  - count : 185
  - lastsampletime : 1300741760533
  - description : Total number of classes unloaded since the Java virtual machine has started execution
  - unit : count
  - name : UnLoadedClassCount
  - starttime : 1300483924127

## Formats for Resource Representation of Log File Details

The Eclipse GlassFish REST interfaces represent resources for log file details in the following formats:

- [JSON](#)
- [XML](#)

### JSON Resource Representation for Log File Details

The general format for the JSON representation of a resource for log file details is as follows:

```
{  
  "records": [  
    record-list  
  ]  
}
```

The replaceable item in this format is the record-list, which is one or more metadata sets separated by a comma (,) that represent the log records in the log file. For the format of each metadata set, see [JSON Representation of a Log Record in a Record List](#).

## JSON Representation of a Log Record in a Record List

The JSON representation of a log record in a record list is as follows:

```
{  
  "recordNumber":record-number,  
  "loggedDateTimeInMS":logged-date,  
  "loggedLevel":"log-level",  
  "productName":"product-name",  
  "loggerName":"logger-name",  
  "nameValuePairs":"_ThreadID=thread-id;_ThreadName=thread-name;",  
  "messageID":"message-id",  
  "Message":"message-text"  
}
```

The replaceable items in this format are as follows:

### **record-number**

A serial number in the form of a decimal integer that uniquely identifies the log record.

### **logged-date**

time when the record was created - a number of milliseconds from the epoch of **1970-01-01T00:00:00Z**.

### **log-level**

The severity level of the message in the log record. For more information, see [Setting Log Levels](#).

### **product-name**

The application that created the log message, for example, **GlassFish 7.0**.

### **logger-name**

The logger name, which is usually a fully qualified name of the Java class owning the logger class that created the log record. For detailed information how to get names of logger classes used in Eclipse GlassFish, see [Listing Loggers](#).

### **thread-id**

The numerical identifier of the thread that created the message.

### **thread-name**

The name of the thread that created the message.

### **message-id**

A unique identifier for the message. For messages from Eclipse GlassFish, this identifier consists of a module code and a numerical value, for example, **CORE5004**. All **SEVERE** and **WARNING** messages and some **INFO** messages from Eclipse GlassFish contain a message identifier. For more information, see the [Eclipse GlassFish Error Message Reference](#).

### **message-text**

The text of the log message.

## Example JSON Resource Representation for Log File Details

This example shows the JSON representation of the resource for log file details. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/management/domain/view-log/details>.

Line breaks and white space are added to enhance readability.

```
{
  "records": [
    {
      "recordNumber":475,
      "loggedDateTimeInMS":1300743782815,
      "loggedLevel":"INFO",
      "productName":"glassfish7",
      "loggerName":"org.glassfish.admingui",
      "nameValuePairs": "_ThreadID=25;_ThreadName=Thread-1;",
      "messageID":"",
      "Message":"Admin Console: Initializing Session Attributes..."
    },
    {
      "recordNumber":474,
      "loggedDateTimeInMS":1300728893368,
      "loggedLevel":"INFO",
      "productName":"glassfish7",
      "loggerName":"
jakarta.enterprise.system.core.com.sun.enterprise.v3.admin.adapter",
      "nameValuePairs":"_ThreadID=238;_ThreadName=Thread-1;",
      "messageID":"",
      "Message":"The Admin Console application is loaded."
    },
    {
      "recordNumber":473,
```

```

    "loggedDateTimeInMS":1300728893367,
    "loggedLevel":"INFO",
    "productName":"glassfish7",
    "loggerName":"jakarta.enterprise.system.core.com.sun.enterprise.v3.server",
    "nameValuePairs":"_ThreadID=238;_ThreadName=Thread-1;",
    "messageID":"CORE10010",
    "Message":" Loading application __admingui done in 40,063 ms"
}
]
}

```

## XML Resource Representation for Log File Details

The general format for the XML representation of a resource for log file details is as follows:

```

<records>
  records
</records>

```

The replaceable item in this format is the records, which is one or more XML elements that represent the log records in the log file. For the format of each element, see [XML Representation of a Log Record](#).

## XML Representation of a Log Record

The XML representation of a log record is as follows:

```

<record loggedDateTimeInMS="logged-date" loggedLevel="log-level"
  loggerName="logger-class-name" messageID="message-id"
  nameValuePairs="_ThreadID=thread-id;_thread-name;" productName="product-name"
  recordNumber="record-number"/>

```

The replaceable items in this format are as follows:

### **logged-date**

time when the record was created - a number of milliseconds from the epoch of **1970-01-01T00:00:00Z**.

### **log-level**

The severity level of the message in the log record. For more information, see [Setting Log Levels](#).

### **logger-class-name**

The fully qualified name of the Java class of the logger class that created the log message. Each component of Eclipse GlassFish provides its own logger class. For detailed information about the

names of logger classes in Eclipse GlassFish, see [Listing Loggers](#).

#### **message-id**

A unique identifier for the message. For messages from Eclipse GlassFish, this identifier consists of a module code and a numerical value, for example, `CORE5004`. All **SEVERE** and **WARNING** messages and some **INFO** messages from Eclipse GlassFish contain a message identifier. For more information, see the [Eclipse GlassFish Error Message Reference](#).

#### **thread-id**

The numerical identifier of the thread that created the message.

#### **thread-name**

The name of the thread that created the message.

#### **product-name**

The application that created the log message, for example, `GlassFish 7.0`.

#### **record-number**

A serial number in the form of a decimal integer that uniquely identifies the log record.

### Example XML Resource Representation for Log File Details

This example shows the XML representation of the resource for log file details. In this example, the DAS is running on the local host and the HTTP port for administration is 4848. The URL to the resource in this example is <http://localhost:4848/management/domain/view-log/details>.

Line breaks and white space are added to enhance readability.

```
<records>
  <record loggedDateTimeInMS="1300743782815" loggedLevel="INFO"
    loggerName="org.glassfish.admingui" messageID=""
    nameValuePairs="_ThreadID=25;_ThreadName=Thread-1;"
    productName="glassfish7" recordNumber="475"/>
  <record loggedDateTimeInMS="1300728893368" loggedLevel="INFO"
    loggerName="jakarta.enterprise.system.core.com.sun.enterprise.v3.admin.adapter"
    messageID="" nameValuePairs="_ThreadID=238;_ThreadName=Thread-1;"
    productName="glassfish7" recordNumber="474"/>
  <record loggedDateTimeInMS="1300728893367" loggedLevel="INFO"
    loggerName="jakarta.enterprise.system.core.com.sun.enterprise.v3.server"
    messageID="core10010" nameValuePairs="_ThreadID=238;_ThreadName=Thread-1;"
    productName="glassfish7" recordNumber="473"/>
</records>
```

## Supported Content Types in Requests to REST Resources

The Eclipse GlassFish REST interfaces support the following types in the content-type header of a client request:

- [JSON](#)
- [XML](#)
- [Form URL encoded](#)

How to specify the type in the content-type header depends on how you are sending the request. For example, if you are using the cURL utility, specify the type through the `-H` option as follows:

- For JSON, specify `-H "Content-type: application/json"`.
- For XML, specify `-H "Content-type: application/xml"`.
- For form URL encoded, specify `-H "Content-type: application/x-www-form-urlencoded"`.

# 3 Administering Domains

This chapter provides procedures for administering domains in the Eclipse GlassFish environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Administering Domains](#)
- [Creating, Logging In To, and Deleting a Domain](#)
- [Starting and Stopping a Domain](#)
- [Configuring a DAS or a Eclipse GlassFish Instance for Automatic Restart](#)
- [Backing Up and Restoring a Domain](#)
- [Re-Creating the Domain Administration Server \(DAS\)](#)
- [Additional Domain Tasks](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About Administering Domains

A domain contains a group of Eclipse GlassFish instances that are administered together. Each domain has a domain administration server (DAS) that hosts administrative applications. These concepts are explained in more detail in the following sections:

- [Eclipse GlassFish Instances](#)
- [Domains for Administering Eclipse GlassFish](#)
- [Domain Administration Server \(DAS\)](#)

## Eclipse GlassFish Instances

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

Eclipse GlassFish instances form the basis of an application deployment.

Whenever a domain is created, Eclipse GlassFish creates a default instance that is named `server`. If a single instance meets your requirements, you can use this instance for deploying applications without the need to administer Eclipse GlassFish instances explicitly. You administer the default instance when you administer its domain.

If you require multiple instances, you must administer the instances explicitly. For more information, see "[Administering Eclipse GlassFish Instances](#)" in Eclipse GlassFish High Availability Administration Guide.

For an instance, you can also create virtual servers. Virtual servers do not span instances. For many purposes, you can use virtual servers instead of multiple instances in operational deployments. Virtual servers enable you to offer, within a single instance, separate domain names, IP addresses, and some administration capabilities to organizations or individuals. To these users, a virtual server behaves like a dedicated web server, but without the hardware and basic web server maintenance.

For more information about virtual servers, see [Administering Virtual Servers](#).

## Domains for Administering Eclipse GlassFish

A domain is an administrative boundary that contains a group of Eclipse GlassFish instances that are administered together. Each instance can belong to only one domain. A domain provides a preconfigured runtime for user applications. Each domain has its own configuration data, log files, and application deployment areas that are independent of other domains. If the configuration is changed for one domain, the configurations of other domains are not affected.

Domains enable different organizations and administrators to share securely a single Eclipse GlassFish installation. Each organization or administrator can administer the instances in a single domain without affecting the instances in other domains.

At installation time, Eclipse GlassFish creates a default domain that is named `domain1`. After installation, you can create additional domains as necessary.

When a domain is created, you are prompted for the administration user name and password. If you accept the default, the user `admin` is created without password. To reset the administration password, see "[To Change an Administration Password](#)" in Eclipse GlassFish Security Guide.

## Domain Administration Server (DAS)

The domain administration server (DAS) is a specially designated Eclipse GlassFish instance that hosts administrative applications. The DAS is similar to any other Eclipse GlassFish instance, except that the DAS has additional administration capabilities. The DAS authenticates the administrator, accepts requests from administration tools, and communicates with other instances in the domain to carry out the requests from administration tools.

Each domain has its own DAS with a unique administration port number. The default administration port is 4848, but a different port can be specified when a domain is created.

The DAS has the master copy of the configuration data for all instances in a domain. If an instance is destroyed, for example, because a host failed, the instance can be re-created from the data in the DAS.

The DAS is the default Eclipse GlassFish instance in a domain and is named `server`. If a single instance meets your requirements, you can use the DAS for deploying applications and for administering the domain.

The graphical Administration Console communicates with a specific DAS to administer the domain that is associated with the DAS. Each Administration Console session enables you to configure and manage only one domain. If you create multiple domains, you must start a separate Administration Console session to manage each domain.

## Creating, Logging In To, and Deleting a Domain

The following topics are addressed here:

- [To Create a Domain](#)
- [To Create a Domain From a Custom Template](#)
- [To List Domains](#)
- [To Log In to a Domain](#)
- [To Delete a Domain](#)

### To Create a Domain

After installing Eclipse GlassFish and creating the default domain (`domain1`), you can create additional domains by using the local `create-domain` subcommand. This subcommand creates the configuration of a domain. Any user who has access to the `asadmin` utility on a given system can create a domain and store the domain configuration in a folder of choice. By default, the domain configuration is created in the default directory for domains. You can override this location to store the configuration elsewhere.

You are required to specify an administrative user when you create a domain, or you can accept the default login identity which is username `admin` with no password.

1. Select a name for the domain that you are creating. You can verify that a name is not already in use by using the `list-domains` subcommand
2. Create a domain by using the `create-domain` subcommand. Information about the options for this subcommand is included in this help page.
3. Type an admin user name and password for the domain. To avoid setting up an admin login, you can accept the default `admin`, with no password. Pressing Return also selects the default.

#### Example 3-1 Creating a Domain

This example creates a domain named `domain1`. When you type the command, you might be prompted for login information.

```
asadmin> create-domain --adminport 4848 domain1
Enter admin user name[Enter to accept default]>
Using port 4848 for Admin.
Default port 8080 for HTTP Instance is in use. Using 1161
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8081 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Default port 8686 for JMX_ADMIN is in use. Using 1162
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=moonbeam.gateway.2wire.net,OU=GlassFish,O=Oracle Corp.,L=Redwood Shores,ST
California,C=US]
Domain domain1 created.
Command create-domain executed successfully.
```

To start the Administration Console in a browser, enter the URL in the following format:

```
http://hostname:5000
```

For this example, the domain's log files, configuration files, and deployed applications now reside in the following directory:

domain-root-dir/[mydomain](#)

#### See Also

You can also view the full syntax and options of the subcommand by typing [asadmin help create-domain](#) at the command line.

## To Create a Domain From a Custom Template

A custom template enables you to customize the configuration of any domain that you create from the template.

1. Create a domain to use as the basis for the template. For more information, see [To Create a Domain](#).
2. Use the [asadmin](#) utility or the Administration Console to configure the domain. Your configuration changes will be included in the template that you create from the domain.
3. Copy the domain's [domain.xml](#) file under a new name to the `as-install/lib/templates` directory. A domain's [domain.xml](#) file is located in the `domain-dir/config` directory.
4. In a plain text editor, edit the file that you copied to replace with tokens values that are to be substituted when a domain is created. Each token is identified as `%%token-name%%`, where `token-name` is

## one of the following names

### ADMIN\_PORT

Represents the port number of the HTTP port or the HTTPS port for administration. This token is replaced with one of the following values in the command to create a domain from the template:

- The value of the `--adminport` option
- The value of the `domain.adminPort` property

### CONFIG\_MODEL\_NAME

Represents the name of the configuration that is created for the domain that is being created. This token is replaced with the string `server-config`.

### DOMAIN\_NAME

Represents the name of the domain that is being created. This token is replaced with the operand of `create-domain` subcommand.

### HOST\_NAME

Represents the name of the host on which the domain is being created. This token is replaced with the fully qualified host name of the host where the domain is being created.

### HTTP\_PORT

Represents the port number of the port that is used to listen for HTTP requests. This token is replaced with one of the following values in the command to create a domain from the template:

- The value of the `--instanceport` option
- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `domain.instancePort` property

### HTTP\_SSL\_PORT

Represents the port number of the port that is used to listen for secure HTTP requests. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `http.ssl.port` property

### JAVA\_DEBUGGER\_PORT

Represents the port number of the port that is used for connections to the [Java Platform Debugger Architecture \(JPDA\)](#) debugger. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `java.debugger.port` property

## JMS\_PROVIDER\_PORT

Represents the port number for the Java Message Service provider. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `jms.port` property

## JMX\_SYSTEM\_CONNECTOR\_PORT

Represents the port number on which the JMX connector listens. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `domain.jmxPort` property

## ORB\_LISTENER\_PORT

Represents the port number of the port that is used for IIOP connections. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `orb.listener.port` property

## ORB\_MUTUALAUTH\_PORT

Represents the port number of the port that is used for secure IIOP connections with client authentication. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `orb.mutualauth.port` property

## ORB\_SSL\_PORT

Represents the port number of the port that is used for secure IIOP connections. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `orb.ssl.port` property

## OSGI\_SHELL\_TELNET\_PORT

Represents 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. This token is replaced with one of the following values in the command to create a domain from the template:

- A value that the `create-domain` subcommand calculates from the value of the `--portbase` option
- The value of the `osgi.shell.telnet.port` property

#### **SERVER\_ID**

Represents the name of the DAS for the domain that is being created. This token is replaced with the string `server`.



For information about how these tokens are used in the default template, examine the `as-install/lib/templates/domain.xml` file.

5. Create the domain that you want to be based on a custom template. In the command to create the domain, pass the name of file that you edited in the previous step as the `--template` option of the `create-domain` subcommand.
6. Before starting the domain, verify that the domain's `domain.xml` file is valid. Use the `verify-domain-xml` subcommand for this purpose. Information about the options for this subcommand is included in the subcommand's help page.

#### See Also

- [To Create a Domain](#)
- [create-domain\(1\)](#)
- [verify-domain-xml\(1\)](#)

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

- `asadmin help create-domain`
- `asadmin help verify-domain-xml`

## To List Domains

Use the `list-domains` subcommand to display a list of domains and their statuses. If the domain directory is not specified, the contents of the domain-root-dir, the default for which is `as-install/domains`, is listed. If there is more than one domain, the domain name must be specified.

To list domains that were created in other directories, specify the `--domaindir` option.

List domains by using the `list-domains` subcommand.

#### Example 3-2 Listing Domains

This example lists the domains in the default domain root directory:

```
asadmin> list-domains
Name: domain1 Status: Running
```

```
Name: domain4 Status: Not Running
Name: domain6 Status: Not Running
Command list-domains executed successfully.
```

## See Also

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

## To Log In to a Domain

All remote subcommands require that credentials be specified in terms of an administration user name and its password. By default, the domain is created with an identity that allows an `asadmin` user to perform administrative operations when no identity is explicitly or implicitly specified.

The default identity is in the form of a user whose name is `admin` and has no password. If you specify no user name on the command line or on prompt, and specify no password in the `--passwordfile` option or on prompt, and you have never logged in to a domain using either the `login` subcommand or the `create-domain` subcommand with the `--savelogin` option, then the `asadmin` utility will attempt to perform a given administrative operation without specifying any identity.

A server (domain) allows administrative operations to be run using this default identity if the following conditions are true:

- The server (domain) uses file realm for authentication of administrative users. If this condition is not true, you will need to specify the user name and password.
- The file realm has one and only one user (what the user name is does not matter). If this condition is not true, you will also need to specify the user name.
- That one user has no password. If this condition is not true, you will need to specify the password.

By default, all of these conditions are true, unless you have created the domain with a specific user name and password. Thus, by default, the only administrative user is `admin` with no password.

Use the `login` subcommand in local mode to authenticate yourself (log in to) a specific domain. After such login, you do not need to specify the administration user or password for subsequent operations on the domain. The `login` subcommand can only be used to specify the administration password. For other passwords that remote subcommands require, use the `--passwordfile` option, or specify the password at the command prompt. You are always prompted for the administration user name and password.

There is no logout subcommand. If you want to log in to another domain, invoke `asadmin login` with new values for `--host` and `--port`.

1. Determine the name of the domain that you are logging in to. To list the existing domains:

```
asadmin list-domains
```

2. Log in to the domain by using the `ologin` command.

#### Example 3-3 Logging In To a Domain on a Remote Machine

This example logs into a domain located on another machine. Options are specified before the `login` subcommand.

```
asadmin> --host foo --port 8282 login
Please enter the admin user name>admin Please enter the admin password>
Trying to authenticate for administration of server at host [foo] and port [8282] ...
Login information relevant to admin user name [admin]
for host [foo] and admin port [8282] stored at [/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this
file will be used by asadmin commands to manage associated domain.
```

#### Example 3-4 Logging In to a Domain on the Default Port of Localhost

This example logs into a domain on `myhost` on the default port. Options are specified before the `login` subcommand.

```
asadmin> --host myhost login
Please enter the admin user name>admin
Please enter the admin password>
Trying to authenticate for administration of server at host [myhost] and port [4848]
...
An entry for login exists for host [myhost] and port [4848], probably from
an earlier login operation.
Do you want to overwrite this entry (y/n)?y
Login information relevant to admin user name [admin] for host [myhost]
and admin port [4848] stored at [/home/joe/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this file will be
used by
asadmin commands to manage associated domain.
```

#### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help login` at the command line. For additional information about passwords, see "Administering Passwords" in Eclipse GlassFish Security Guide.

## To Delete a Domain

Use the `delete-domain` subcommand to delete an existing domain from a server. Only the root user

or the operating system user who is authorized to administer the domain can run this subcommand.

## Before You Begin

A domain must be stopped before it can be deleted.

1. List domains by using the [list-domains](#) subcommand.
2. If necessary, notify domain users that the domain is being deleted.
3. Ensure that the domain you want to delete is stopped. If needed, see [To Stop a Domain](#).
4. Delete the domain by using the [delete-domain](#) subcommand.

## Example 3-5 Deleting a Domain

This example deletes a domain named `domain1` from the location specified.

```
asadmin> delete-domain --domaindir ..\domains domain1
Domain domain1 deleted.
Command delete-domain executed successfully.
```

## See Also

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

# Starting and Stopping a Domain

The following topics are addressed here:

- [To Start a Domain](#)
- [To Stop a Domain](#)
- [To Restart a Domain](#)

## To Start a Domain

When you start a domain or server, the domain administration server (DAS) is started. After startup, the DAS runs constantly, listening for and accepting requests.

If the domain directory is not specified, the domain in the default domain root directory is started. If there are two or more domains, the `domain_name` operand must be specified. Each domain must be started separately.



For Microsoft Windows, you can use an alternate method to start a domain. From

the Windows Start menu, select the command for your distribution of Eclipse GlassFish:

- If you are using the Full Platform, select Programs > Eclipse GlassFish > Start Admin Server.
- If you are using the Web Profile, select Programs > Eclipse GlassFish Web Profile > Start Admin Server.

This subcommand is supported in local mode only.

Start a domain by using the `start-domain` subcommand.

#### Example 3-6 Starting a Domain

This example starts `domain2` in the default domain directory.

```
asadmin> start-domain domain2
```

If there is only one domain, you can omit the domain name. If you do not include the password, you might be prompted to supply it.

```
Name of the domain started: [domain1] and its location:  
[C:\prelude\v3_prelude_release\distributions\web\target\glassfish  
domains\domain1].  
Admin port for the domain: [4848].
```

#### See Also

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

## To Stop a Domain

Stopping a domain or server shuts down its domain administration server (DAS). When stopping a domain, the DAS stops accepting new connections and then waits for all outstanding connections to complete. This shutdown process takes a few seconds. While the domain is stopped, the Administration Console and most of the `asadmin` subcommands cannot be used. This subcommand is particularly useful in stopping a runaway server. For more controlled situations, you can use the `restart-domain` subcommand.

For Microsoft Windows, you can use an alternate method to stop a domain. From the Start menu, select the command for your distribution of Eclipse GlassFish:



- If you are using the Full Platform, select Programs > Eclipse GlassFish > Stop Admin Server.

- If you are using the Web Profile, select Programs > Eclipse GlassFish Web Profile > Stop Admin Server.
1. If necessary, notify users that you are going to stop the domain.
  2. Stop the domain by using the `stop-domain` subcommand.

#### Example 3-7 Stopping a Domain (or Server)

This example stops `domain1` in the default directory, where `domain1` is the only domain present in the directory.

```
asadmin> stop-domain
Waiting for the domain to stop .....
Command stop-domain executed successfully.
```

#### See Also

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

## To Restart a Domain

Use the `restart-domain` subcommand in remote mode to restart the Domain Administration Server (DAS) of the specified host. When restarting a domain, the DAS stops accepting new connections and then waits for all outstanding connections to complete. This shutdown process takes a few seconds. Until the domain has restarted, the Administration Console and most of the `asadmin` subcommands cannot be used.

This subcommand is particularly useful for environments where the server machine is secured and difficult to get to. With the right credentials, you can restart the server from a remote location as well as from the same machine.

If the server will not restart, use the `stop-domain` subcommand followed by the `start-domain` subcommand.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Restart the domain by using the `restart-domain` subcommand.

#### Example 3-8 Restarting a Domain (or Server)

This example restarts `mydomain4` in the default directory.

```
asadmin> restart-domain mydomain4
Waiting for the domain to restart .....
Command restart-domain executed successfully.
```

### Example 3-9 Restarting a Domain in a Browser

This example invokes the `restart-domain` subcommand in a browser.

```
http://yourhost:4848/__asadmin/restart-domain
```

#### See Also

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

## Configuring a DAS or a Eclipse GlassFish Instance for Automatic Restart

Use the `create-service` subcommand in local mode to configure your system to automatically restart a domain administration server (DAS) or a Eclipse GlassFish instance. Eclipse GlassFish enables you to configure a DAS or an instance for automatic restart on the following operating systems:

- Windows
- Linux
- Oracle Solaris

To ensure that automatic restart functions correctly on Windows, you must prevent service shutdown when a user logs out.

The following topics are addressed here:

- [To Configure a DAS or an Instance for Automatic Restart on Windows](#)
- [To Configure a DAS or an Instance for Automatic Restart on Linux](#)
- [To Configure a DAS or an Instance for Automatic Restart on Oracle Solaris](#)
- [To Prevent Service Shutdown When a User Logs Out on Windows](#)

### To Configure a DAS or an Instance for Automatic Restart on Windows

On Windows systems, the `create-service` subcommand creates a Windows service to represent the DAS or instance. The service is created in the disabled state. After this subcommand creates the service, you must use the Windows Services Manager or the Windows Services Wrapper to start, stop, uninstall, or install the service. To administer the service from the Windows command line, use the `sc.exe` tool.

This subcommand must be run as the OS-level administrator user.

1. Create the service by using the `create-service` subcommand.
2. After the service is created, start the service by using the Windows Services Manager or the Windows Services Wrapper.

For example, to start the service for the default domain by using the `sc.exe` tool, type:

```
C:\> sc start domain1
```

If you are using the `sc.exe` tool to administer the service, use the tool as follows:

- To obtain information about the service, use the `sc query` command.
- To stop the service, use the `sc stop` command.
- To uninstall the service, use the `sc delete` command.

#### Example 3-10 Creating a Service to Restart a DAS Automatically on Windows

This example creates a service for the default domain on a system that is running Windows.

```
asadmin> create-service
Found the Windows Service and successfully uninstalled it.
The Windows Service was created successfully. It is ready to be started. Here are
the details:
ID of the service: domain1
Display Name of the service:domain1 Eclipse GlassFish
Domain Directory: C:\glassfish7\glassfish\domains\domain1
Configuration file for Windows Services Wrapper: C:\glassfish7\glassfish\domains\
domain1\bin\domain1Service.xml
The service can be controlled using the Windows Services Manager or you can use the
Windows Services Wrapper instead:
Start Command: C:\glassfish7\glassfish\domains\domain1\bin\domain1Service.exe start
Stop Command: C:\glassfish7\glassfish\domains\domain1\bin\domain1Service.exe stop
Uninstall Command: C:\glassfish7\glassfish\domains\domain1\bin\domain1Service.exe
uninstall
Install Command: C:\glassfish7\glassfish\domains\domain1\bin\domain1Service.exe
install

This message is also available in a file named PlatformServices.log in the domain's
root directory
Command create-service executed successfully.
```

#### Example 3-11 Querying the Service to Restart a DAS Automatically on Windows

This obtains information about the service for the default domain on a system that is running Windows.

```
C:\> sc query domain1
```

```
SERVICE_NAME: domain1
  TYPE          : 10  WIN32_OWN_PROCESS
  STATE         : 1   STOPPED
  WIN32_EXIT_CODE : 1077 (0x435)
  SERVICE_EXIT_CODE : 0 (0x0)
  CHECKPOINT    : 0x0
  WAIT_HINT     : 0x0
```

## To Configure a DAS or an Instance for Automatic Restart on Linux

On Linux systems, the `create-service` subcommand creates a System-V-style initialization script `/etc/init.d/GlassFish_domain-or-instance-name` and installs a link to this script in any `'/etc/rc`N.d`` directory that is present, where N is 0, 1, 2, 3, 4, 5, 6, and 5. After this subcommand creates the script, you must use this script to start, stop, or restart the domain or instance.

The script automatically restarts the domain or instance only during a reboot. If the domain or instance is stopped, but the host remains running, the domain or instance is not restarted automatically. To restart the domain or instance, you must run the script manually.

You might no longer require the domain or instance to be automatically restarted during a reboot. In this situation, use the operating system to delete the initialization script and the link to the script that the `create-service` subcommand creates.

The `create-service` subcommand must be run as the OS-level root user.

Create the service by using the `create-service` subcommand.

### Example 3-12 Creating a Service to Restart a DAS Automatically on Linux

This example creates a service for the default domain on a system that is running Linux.

```
asadmin> create-service
Found the Linux Service and successfully uninstalled it.
The Service was created successfully. Here are the details:
Name of the service:domain1
Type of the service:Domain
Configuration location of the service:/etc/init.d/GlassFish_domain1
User account that will run the service: root
You have created the service but you need to start it yourself.
Here are the most typical Linux commands of interest:

* /etc/init.d/GlassFish_domain1 start
* /etc/init.d/GlassFish_domain1 stop
* /etc/init.d/GlassFish_domain1 restart
```

For your convenience this message has also been saved to this file:  
`/export/glassfish7/glassfish/domains/domain1/PlatformServices.log`

```
Command create-service executed successfully.
```

## To Configure a DAS or an Instance for Automatic Restart on Oracle Solaris

On Oracle Solaris systems, the `create-service` subcommand creates an Oracle Solaris Service Management Facility (SMF) service that restarts a DAS or an instance. The service grants to the process the privileges of the user that runs the process. When you create an SMF service, the default user is the superuser. If you require a different user to run the process, specify the user in `method_credential`.

If your process is to bind to a privileged port of Oracle Solaris, the process requires the `net_privaddr` privilege. The privileged ports of the Oracle Solaris operating system have port numbers less than 1024.

To determine if a user has the `net_privaddr` privilege, log in as that user and type the command `ppriv -l | grep net_privaddr`.

After you create and enable the SMF service, if the domain or instance is stopped, SMF restarts it.

### Before You Begin

To run the `create-service` subcommand, you must have `solaris.smf.*` authorization. For information about how to set the authorizations, see the `useradd(1M)` man page and the `usermod(1M)` man page. You must also have write permission in the directory tree: `/var svc/manifest/application/SUNWappserver`. Usually, the superuser has both of these permissions. Additionally, Oracle Solaris administration commands such as `svccfg`, `svcs`, and `auths` must be available in the PATH.

If a particular Eclipse GlassFish domain or instance should not have default user privileges, modify the manifest of the service and reimport the service.

1. Create the service by using the `create-service` subcommand.
2. After the service is created, enable the service by using the `svacdm enable` command.

For example, to enable the SMF service for the default domain, type:

```
svacdm enable /appserver/domains/domain1
```

### Example 3-13 Creating a Service to Restart a Domain Automatically on Oracle Solaris

This example creates a service for the default domain on a system that is running Oracle Solaris.

```
asadmin> create-service
The Service was created successfully. Here are the details:
Name of the service:application/GlassFish/domain1
Type of the service:Domain
```

```
Configuration location of the service:/home/gfuser/glassfish-installations  
/glassfish7/glassfish/domains  
Manifest file location on the system:/var/svc/manifest/application  
/GlassFish/domain1_home_gfuser_glassfish-installations_glassfish7  
_glassfish_domains/Domain-service-smf.xml.  
You have created the service but you need to start it yourself.  
Here are the most typical Solaris commands of interest:  
* /usr/bin/svcs -a | grep domain1 // status  
* /usr/sbin/svcadm enable domain1 // start  
* /usr/sbin/svcadm disable domain1 // stop  
* /usr/sbin/svccfg delete domain1 // uninstall  
Command create-service executed successfully
```

## See Also

For information about administering the service, see the following Oracle Solaris documentation:

- "[Managing Services \(Overview\)](#)" in System Administration Guide: Basic Administration
- "[Managing Services \(Tasks\)](#)" in System Administration Guide: Basic Administration
- [auths\(1\)](#)
- [svcs\(1\)](#)
- [svcadm\(1M\)](#)
- [svccfg\(1M\)](#)
- [useradd\(1M\)](#)
- [usermod\(1M\)](#)
- [rbac\(5\)](#)
- [smf\\_security\(5\)](#)

## To Prevent Service Shutdown When a User Logs Out on Windows

By default, the Java Virtual Machine (VM) receives signals from Windows that indicate that Windows is shutting down, or that a user is logging out of Windows, which causes the system to shut itself down cleanly. This behavior causes the Eclipse GlassFish service to shut down. To prevent the service from shutting down when a user logs out, you must set the `-Xrs` Java VM option (<https://github.com/eclipse-ee4j/glassfishdocumentation>).

1. Ensure that the DAS is running.
2. Set the `-Xrs` Java VM option for the DAS. Use the `create-jvm-options` subcommand for this purpose.

```
asadmin> create-jvm-options -Xrs
```

3. Set the `-Xrs` Java VM option for the Java VM within which the `asadmin` utility runs. To set this option, edit the `asadmin.bat` file to add the `-Xrs` option to the line that runs the `admin-cli.jar` file.

a. In the `as-install\bin\asadmin.bat` file, edit the line to read as follows:

```
%JAVA% -Xrs -jar "%~dp0..\modules\admin-cli.jar" %*
```

b. In the `as-install-parent\bin\asadmin.bat` file, edit the line to read as follows:

```
%JAVA% -Xrs -jar "%~dp0..\glassfish\modules\admin-cli.jar" %*
```

4. If the Eclipse GlassFish service is running, restart the service for your changes to take effect.

## Backing Up and Restoring a Domain

The following topics are addressed here:

- [To Back Up a Domain](#)
- [To Restore a Domain](#)
- [To List Domain Backups](#)

### To Back Up a Domain

Use the `backup-domain` subcommand in local mode to make a backup of a specified domain.

When you use the `backup-domain` subcommand, Eclipse GlassFish creates a ZIP file backup of all the files and subdirectories in the domain's directory, `domain-root-dir/domain-dir`, except for the `backups` subdirectory.

The `backup-domain` subcommand provides several options to meet particular needs, including:

- `--backupdir` to specify a directory in which to store the backup instead of the default `domain-root-dir/domain-dir/backups`.
- `--description` to provide a description of the backup to be stored in the backup itself.

1. Ensure that the domain is stopped .

The `backup-domain` subcommand operates only when the domain is stopped.

2. Back up the domain by using the `backup-domain` subcommand.

3. Restore the domain to its previous state, if necessary.

Start or resume the domain.

## Example 3-14 Backing Up the Default Domain

This example makes a backup of the default domain, `domain1`, storing the backup file in `/net/backups.example.com/glassfish`:

```
asadmin> backup-domain --backupdir /net/backups.example.com/glassfish domain1
Backed up domain1 at Mon Jan 17 08:16:22 PST 2011.
Command backup-domain executed successfully
```

### See Also

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

## To Restore a Domain

Use the `restore-domain` subcommand in local mode to use a backup file to restore the files and subdirectories in a specified domain's directory.

The `restore-domain` subcommand can use backup files created by the `backup-domain` subcommand and by automatic backup configurations, both full backups and configuration-only backups. Automatic backup configurations are available only in Eclipse GlassFish.

1. If necessary, notify domain users that the domain is being restored from backup.
2. Ensure that the domain is stopped.

The `restore-domain` subcommand operates only when the domain is stopped.

To determine whether the domain is running, use the `list-domains` subcommand, as described in [To List Domains](#).

To stop the domain, use the `stop-domain` subcommand as described in [To Stop a Domain](#).

3. Restore backup files for a domain by using the `restore-domain` subcommand.
4. Verify that the restore has succeeded.
5. If necessary, notify users that the domain has been restored and is available.

## Example 3-15 Restoring the Default Domain

This example restores files for the default domain, `domain1`, from the most recent backup stored in a specified backup directory:

```
asadmin> restore-domain --backupdir /net/backups.example.com/glassfish domain1
Restored the domain (domain1) to /home/user1/glassfish7/glassfish/domains/domain1
Command restore-domain executed successfully.
```

## See Also

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

## To List Domain Backups

Use the `list-backups` subcommand in local mode to display information about backups of a specified domain stored in a specified backup directory.

The `list-backups` subcommand provides several options to meet particular needs, including `--backupdir` to specify a directory where backups are stored instead of the default `domain-dir/backups`.

List backups by using the `list-backups` subcommand.

### Example 3-16 Listing Backups of the Default Domain

This example lists the backups of the default domain, `domain1`, that are stored in the `/net/backups.example.com/glassfish` directory:

```
asadmin> list-backups --backupdir /net/backups.example.com/glassfish domain1
CONFIG      USER    BACKUP DATE          FILENAME
           user1  Mon Jan 17 08:16:22 PST 2011  domain1_2011_01_17_v00001.zip
monthly-full user1  Wed Dec 01 00:00:00 PST 2010  domain1_2010_12_01_v00001.zip
monthly-full user1  Sat Jan 01 00:00:03 PST 2011  domain1_2011_01_01_v00001.zip
monthly-full user1  Tue Feb 01 00:00:01 PST 2011  domain1_2011_02_01_v00001.zip
Command list-backups executed successfully.
```

Note that this listing includes backups created automatically by a backup configuration. This feature is available only in Eclipse GlassFish.

## See Also

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

## Re-Creating the Domain Administration Server (DAS)

For mirroring purposes, and to provide a working copy of the DAS, you must have:

- One host (`olddashost`) that contains the original DAS.
- A second host (`apphost`) that contains a cluster with server instances running applications and catering to clients. The cluster is configured using the DAS on the first host.
- A third host (`newdashost`) where the DAS needs to be re-created in a situation where the first

host crashes or is being taken out of service.

 You must maintain a backup of the DAS from the first host using the `obackup-domain` subcommand as described in [To Back Up a Domain](#). You can automatically maintain a backup of the DAS using the automatic backups feature of Eclipse GlassFish.

 Eclipse GlassFish includes `asadmin` subcommands that simplify this procedure. If you are using Eclipse GlassFish, see [To Migrate the DAS](#).

## To Migrate the DAS

The following steps are required to migrate the DAS from the first host (olddashost) to the third host (newdashost).

1. Install Eclipse GlassFish on newdashost just as it was installed on olddashost. This is required so that the DAS can be properly restored on newdashost without causing path conflicts.
2. Use the `restore-domain` subcommand to restore the latest backup file onto newdashost. For example:

```
asadmin> restore-domain --backupdir /net/backups.example.com/glassfish
```

This example assumes that backups are stored in a network-accessible location. If this is not the case, manually copy the latest backup file from offline storage to a directory on newdashost. You can backup any domain. However, while re-creating the domain, the domain name should be same as the original.

3. Stop the domain on olddashost, if it is running.
4. Start the domain on newdashost by using the `start-domain` subcommand. For example:

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

5. If the domain on olddashost was centrally administered, set up centralized administration on newdashost. See "[Enabling Centralized Administration of Eclipse GlassFish Instances](#)" in Eclipse GlassFish High Availability Administration Guide for instructions.
6. Verify that instances on other hosts are visible to the new DAS on newdashost:

```
asadmin> list-instances --long
```

7. Change the DAS host values for properties under the node on apphost. In the file `as-install/nodes/node-name/agent/config/das.properties` file, change the `agent.das.host` property value to refer to newdashost instead of olddasnost.

8. Use the new DAS to restart clusters and standalone instances on apphost: Restarting the clustered and standalone instances on apphost triggers their recognition of the new DAS on newdashost.
  - a. Use the `list-clusters` subcommand to list the clusters in the domain.
  - b. Use the `stop-cluster` subcommand to stop each cluster.
  - c. Use the `list-instances` subcommand to list the instances in the domain.
  - d. Use the `restart-instance` subcommand to restart each standalone instance.
  - e. Use the `start-cluster` subcommand to start each cluster. If the domain does not use centralized administration, use the `start-local-instance` subcommand to start the cluster instances on apphost.
9. Verify that instances on apphost are running:

```
asadmin> list-instances --long
```

10. Decommission and discontinue use of the DAS on olddashost.

## Additional Domain Tasks

The following topics are addressed here:

- [To Display Domain Uptime](#)
- [To Switch a Domain to Another Supported Java Version](#)
- [To Change the Administration Port of a Domain](#)

### To Display Domain Uptime

Use the `uptime` subcommand in remote mode to display the length of time that the domain administration server (DAS) has been running since it was last started.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Display uptime by using the `uptime` subcommand.

#### Example 3-17 Displaying the DAS Uptime

This example displays the length of time that the DAS has been running.

```
asadmin> uptime
Uptime: 1 Weeks, 4 days, 0 hours, 17 minutes, 14 seconds, Total milliseconds:
951434595
Command uptime executed successfully.
```

## See Also

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

## To Switch a Domain to Another Supported Java Version

Eclipse GlassFish 7 requires Java SE 11 as the underlying virtual machine for the Java platform (Java Virtual Machine or JVM machine).



Do not downgrade to an earlier Java version after a domain has been created with a newer JVM machine. If you must downgrade your JVM machine, downgrade it only for individual domains.

1. If you have not already done so, download the desired Java SDK (not the JRE) and install it on your system. The Java SDK can be downloaded from the [Java SE RI Downloads page](#).
2. Start the domain for which you are changing the JDK. Use the following format:

```
as-install/bin/asadmin start-domain domain-name
```

For a valid JVM installation, locations are checked in the following order:

1. `domain.xml` (`java-home` inside `java-config`)
2. `asenv.conf` (setting `AS_JAVA="path to java home"`) If a legal JDK is not found, a fatal error occurs and the problem is reported back to you.
3. If necessary, change the JVM machine attributes for the domain. In particular, you might need to change the `JAVA_HOME` environment variable. For example, to change the `JAVA_HOME` variable, type:

```
as-install/bin/asadmin set "server.java-config.java-home=path-to-java-home"
```

## To Change the Administration Port of a Domain

Use the `set` subcommand in remote mode to change the administration port of a domain.

The HTTP port or the HTTPS port for administration of a domain is defined by the `--adminport` option of the `create-domain` subcommand when the domain is created. If this port must be reallocated for another purpose, change the port on which the DAS listens for administration requests.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Set the port number to its new value. Use the `set` subcommand for this purpose.

```
$ asadmin set  
server-config.network-config.network-listeners.network-listener.admin-  
listener.port=new-port-number
```

The new-port-number is the new value that you are setting for the port number.



After you set the port number to its new value, running the [list-domains](#) subcommand incorrectly reports that the DAS is not running. The [list-domains](#) subcommand reports the correct state again only after you stop and restart the domain as explained in the steps that follow.

3. Stop the domain, specifying the host on which the DAS is running and the old administration port number of the domain. You must specify the old port number because the DAS is still listening for administration requests on this port. If you omit the port number, the command fails because the [stop-domain](#) subcommand attempts to contact the DAS through the new port number.



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 domain, see the [ostop-domain\(1\)](#) help page.

```
$ asadmin --host host-name --port old-port-number stop-domain
```

#### host-name

The name of the host on which the DAS is running. If you run the [stop-domain](#) subcommand on the host where the DAS is running, you must specify the actual host name and not [localhost](#). If you specify [localhost](#), the [stop-domain](#) subcommand fails.

#### old-port-number

The value of administration port number of the domain before you changed it in the preceding step.

4. Start the domain.



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 domain, see the [ostart-domain\(1\)](#) help page.

```
$ start-domain [domain-name]
```

The domain-name is the name of the domain to start. If only one domain subdirectory is contained in the [domains](#) directory, you may omit this option.

#### Example 3-18 Changing the Administration Port of a Domain

This example changes the administration port of the domain `domain1` from 4848 to 4849. The DAS is running on the host `xk01.example.com`.

```
$ asadmin set  
server-config.network-config.network-listeners.network-listener.admin-  
listener.port=4849  
server-config.network-config.network-listeners.network-listener.admin-  
listener.port=4849  
Command set executed successfully.  
$ asadmin --host xk01.example.com --port 4848 stop-domain  
Waiting for the domain to stop ....  
Command stop-domain executed successfully.  
$ asadmin start-domain  
Waiting for domain1 to start .....  
Successfully started the domain : domain1  
domain Location: /export/glassfish7/glassfish/domains/domain1  
Log File: /export/glassfish7/glassfish/domains/domain1/logs/server.log  
Admin Port: 4849  
Command start-domain executed successfully.
```

## See Also

- [create-domain\(1\)](#)
- [set\(1\)](#)
- [start-domain\(1\)](#)
- [stop-domain\(1\)](#)

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

- `asadmin help create-domain`
- `asadmin help set`
- `asadmin help start-domain`
- `asadmin help stop-domain`

# 4 Administering the Virtual Machine for the Java Platform

This chapter provides procedures for administering the Virtual Machine for the Java platform (Java Virtual Machine) or JVM machine) in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [Administering JVM Options](#)
- [Administering the Profiler](#)

Instructions for accomplishing these tasks by using the Administration Console are contained in the Administration Console online help.

## Administering JVM Options

The Java Virtual Machine is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The virtual machine translates the Java byte codes into the native instructions of the host machine. Eclipse GlassFish, being a Java process, requires a virtual machine to run and support the Java applications running on it. JVM settings are part of an Eclipse GlassFish configuration.

The following topics are addressed here:

- [To Create JVM Options](#)
- [To List JVM Options](#)
- [To Delete JVM Options](#)
- [To Generate a JVM Report](#)

### To Create JVM Options

Use the `create-jvm-options` subcommand in remote mode to create JVM options in the Java configuration or the profiler elements of the `domain.xml` file. If JVM options are created for a profiler, these options are used to record the settings that initiate the profiler.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create JVM options by using the `create-jvm-options` subcommand.

To create more than one JVM option, use a colon (:) to separate the options. If the JVM option itself contains a colon (:), use the backslash () to offset the colon delimiter.

Information about properties for the subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish. See [To Restart a Domain](#).

#### Example 4-1 Creating JVM Options

This example sets multiple Java system properties.

```
asadmin> create-jvm-options -Dunixlocation=/root/example:  
-Dvariable=\$HOME:  
-Dwindowslocation=d:\\\\\\sun\\\\appserver:  
-Doption1=value1  
created 4 option(s)  
Command create-jvm-options executed successfully.
```

#### See Also

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

## To List JVM Options

Use the `list-jvm-options` subcommand in remote mode to list the existing JVM options.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List JVM options by using the `list-jvm-options` subcommand.

#### Example 4-2 Listing JVM Options

This example lists all JVM options.

```
asadmin> list-jvm-options  
-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf  
-XX: LogVMOutput  
-XX: UnlockDiagnosticVMOptions  
-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory  
-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks  
-XX:NewRatio=2  
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy  
-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver  
-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks  
-client  
-Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext${path.separator}${com.sun.aas.javaRoot}/jre/lib/ext${path.separator}${com.sun.aas.instanceRoot}/lib/ext${path.separator}${com.sun.aas.derbyRoot}/lib  
-Xmx512m
```

```
-XX:LogFile=${com.sun.aas.instanceRoot}/logs/jvm.log
Command list-jvm-options executed successfully.
```

## See Also

You can also view the full syntax and options of the subcommand by typing [asadmin help list-jvm-options](#) at the command line.

## To Delete JVM Options

Use the [delete-jvm-options](#) subcommand in remote mode to delete JVM options from the Java configuration or profiler elements of the [domain.xml](#) file.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List JVM options by using the [list-jvm-options](#) subcommand.
3. If necessary, notify users that the JVM option is being deleted.
4. Delete JVM options by using the [delete-jvm-options](#) subcommand.

To remove more than one JVM option, use a colon (:) to separate the options. If the JVM option itself contains a colon, use the backslash (\) to offset the colon delimiter.

5. To apply your changes, restart Eclipse GlassFish. See [To Restart a Domain](#).

### Example 4-3 Deleting a JVM Option

This example removes a single JVM option.

```
asadmin> delete-jvm-options -Dopt1=A

deleted 1 option(s)
Command delete-jvm-options executed successfully.
```

### Example 4-4 Deleting Multiple JVM Options

This example removes multiple JVM options.

```
asadmin> delete-jvm-options -Doption1=-value1:-Dvariable=\$HOME
deleted 2 option(s)
Command delete-jvm-options executed successfully.
```

## See Also

You can also view the full syntax and options of the subcommand by typing [asadmin help delete-jvm-options](#) at the command line.

## To Generate a JVM Report

Use the `generate-jvm-report` subcommand in remote mode to generate a JVM report showing the threads (dump of a stack trace), classes, memory, and loggers for a specified instance, including the domain administration server (DAS). You can generate the following types of reports: summary (default), class, thread, log.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Generate the report by using the `generate-jvm-report` subcommand.

### Example 4-5 Generating a JVM Report

This example displays summary information about the threads, classes, and memory.

```
asadmin> generate-jvm-report --type summary
Operating System Information:
Name of the Operating System: Windows XP
Binary Architecture name of the Operating System: x86, Version: 5.1
Number of processors available on the Operating System: 2
System load on the available processors for the last minute: NOT_AVAILABLE.
(Sum of running and queued runnable entities per minute).

.
.

user.home = C:\Documents and Settings\Jennifer
user.language = en
user.name = Jennifer
user.timezone = America/New_York
user.variant =
variable = \$HOME
web.home = C:\Preview\v3_Preview_release\distributions\web\target\
glassfish\modules\web
Command generate-jvm-report executed successfully.
```

### See Also

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

## Administering the Profiler

A profiler generates information used to analyze server performance.

The following topics are addressed here:

- [To Create a Profiler](#)
- [To Delete a Profiler](#)

## To Create a Profiler

A server instance is tied to a particular profiler by the profiler element in the Java configuration. If JVM options are created for a profiler, the options are used to record the settings needed to activate a particular profiler. Use the `create-profiler` subcommand in remote mode to create the profiler element in the Java configuration.

Only one profiler can exist. If a profiler already exists, you receive an error message that directs you to delete the existing profiler before creating a new one.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a profiler by using the `create-profiler` subcommand.

Information about properties for the subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 4-6 Creating a Profiler

This example creates a profiler named `sample_profiler`.

```
asadmin> create-profiler --classpath=/home/appserver/ --nativelibrarypath=/u/home/lib  
--enabled=false --property=defaultuser=admin:password=adminadmin sample_profiler  
Command create-profiler executed successfully.
```

### See Also

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

## To Delete a Profiler

Use the `delete-profiler` subcommand in remote mode to delete the profiler element from the Java configuration. You can then create a new profiler.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Delete the profiler by using the `delete-profiler` subcommand.
3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 4-7 Deleting a Profiler

This example deletes the profiler named `sample_profiler`.

```
asadmin> delete-profiler sample_profiler  
Command delete-profiler executed successfully.
```

#### See Also

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

# 5 Administering Thread Pools

This chapter provides procedures for administering thread pools in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Thread Pools](#)
- [Configuring Thread Pools](#)

Instructions for accomplishing these tasks by using the Administration Console are contained in the Administration Console online help.

## About Thread Pools

The Virtual Machine for the Java platform (Java Virtual Machine) or JVM machine) can support many threads of execution simultaneously. To help performance, Eclipse GlassFish maintains one or more thread pools. It is possible to assign specific thread pools to connector modules, to network listeners, or to the Object Request Broker (ORB).

One thread pool can serve multiple connector modules and enterprise beans. Request threads handle user requests for application components. When Eclipse GlassFish receives a request, it assigns the request to a free thread from the thread pool. The thread executes the client's requests and returns results. For example, if the request needs to use a system resource that is currently busy, the thread waits until that resource is free before allowing the request to use that resource.

## Configuring Thread Pools

You can specify the minimum and maximum number of threads that are reserved for requests from applications. The thread pool is dynamically adjusted between these two values.

The following topics are addressed here:

- [To Create a Thread Pool](#)
- [To List Thread Pools](#)
- [To Update a Thread Pool](#)
- [To Delete a Thread Pool](#)

### To Create a Thread Pool

Use the `create-threadpool` subcommand in remote mode to create a thread pool.

The minimum thread pool size that is specified signals the server to allocate at least that many threads in reserve for application requests. That number is increased up to the maximum thread pool size that is specified. Increasing the number of threads available to a process allows the process to respond to more application requests simultaneously.

If one resource adapter or application occupies all the Eclipse GlassFish threads, thread starvation might occur. You can avoid this by dividing the Eclipse GlassFish threads into different thread pools.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a new thread pool by using the `create-threadpool` subcommand.

Information about options for the subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).



Restart is not necessary for thread pools used by the web container.

#### Example 5-1 Creating a Thread Pool

This example creates `threadpool-1`.

```
asadmin> create-threadpool --maxthreadpools 100  
--minthreadpools 20 --idletimeout 2 --workqueues 100 threadpool-1  
Command create-threadpool executed successfully
```

#### See Also

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

## To List Thread Pools

Use the `list-threadpools` subcommand in remote mode to list the existing thread pools.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing thread pools by using the `list-threadpools` subcommand.

#### Example 5-2 Listing Thread Pools

This example lists the existing thread pools.

```
asadmin> list-threadpools  
threadpool-1
```

```
Command list-threadpools executed successfully
```

## See Also

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

## To Update a Thread Pool

Use the `set` subcommand to update the values for a specified thread pool.

1. List the existing thread pools by using the `list-threadpools` subcommand.
2. Modify the values for a thread pool by using the `set` subcommand.

The thread pool is identified by its dotted name.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).



Restart is not necessary for thread pools used by the web container.

### Example 5-3 Updating a Thread Pool

This example sets the `max-thread-pool-size` from its previous value to 8. [source]

```
asadmin> set server.thread-pools.thread-pool.http-thread-pool.max-thread-pool-size=8
Command set executed successfully
```

## See Also

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

## To Delete a Thread Pool

Use the `delete-threadpool` subcommand in remote mode to delete an existing thread pool. Deleting a thread pool will fail if that pool is referenced by a network listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing thread pools by using the `list-threadpools` subcommand.
3. Delete the specified thread pool by using the `delete-threadpool` subcommand.
4. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).



Restart is not necessary for thread pools used by the web container.

#### Example 5-4 Deleting a Thread Pool

This example deletes **threadpool-1**.

```
asadmin> delete-threadpool threadpool-1  
Command delete-threadpool executed successfully
```

#### See Also

You can also view the full syntax and options of the subcommand by typing **asadmin help delete-threadpool** at the command line.

# 6 Administering Web Applications

This chapter explains how to administer web applications in the Eclipse GlassFish 7 environment.

The following topics are addressed here:

- [Invoking a Servlet by Alternate Means](#)
- [Changing Log Output for a Servlet](#)
- [Defining Global Features for Web Applications](#)
- [Redirecting a URL](#)
- [Administering mod\\_jk](#)
- [Administering mod\\_proxy\\_ajp](#)

Instructions for accomplishing some of these tasks by using the Administration Console are contained in the Administration Console online help.

## Invoking a Servlet by Alternate Means

You can call a servlet deployed to Eclipse GlassFish by using a URL in a browser or embedded as a link in an HTML or JSP file. The format of a servlet invocation URL is as follows:

```
http://server:port/context-root/servlet-mapping?name=value
```

The following table describes each URL section.

Table 6-1 URL Fields for Servlets Within an Application

URL element	Description
server`:`port	The IP address (or host name) and optional port number.  To access the default web module for a virtual server, specify only this URL section. You do not need to specify the context-root or servlet-name unless you also wish to specify name-value parameters.
context-root	For an application, the context root is defined in the <code>context-root</code> element of the <code>application.xml</code> , <code>sun-application.xml</code> , or <code>sun-web.xml</code> file. For an individually deployed web module, the context root is specified during deployment.  For both applications and individually deployed web modules, the default context root is the name of the WAR file minus the <code>.war</code> suffix.
servlet-mapping	The <code>servlet-mapping</code> as configured in the <code>web.xml</code> file.

URL element	Description
?`name=value...`	Optional request parameters.

### Example 6-1 Invoking a Servlet With a URL

In this example, `localhost` is the host name, `MortPages` is the context root, and `calcMortgage` is the servlet mapping.

```
http://localhost:8080/MortPages/calcMortgage?rate=8.0&per=360&bal=180000
```

### Example 6-2 Invoking a Servlet From Within a JSP File

To invoke a servlet from within a JSP file, you can use a relative path. For example:

```
<jsp:forward page="TestServlet"/><jsp:include page="TestServlet"/>
```

## Changing Log Output for a Servlet

`ServletContext.log` messages are sent to the server log. By default, the `System.out` and `System.err` output of servlets are sent to the server log. During startup, server log messages are echoed to the `System.err` output. Also by default, there is no Windows-only console for the `System.err` output.

You can change these defaults using the Administration Console Write to System Log box. If this box is checked, `System.out` output is sent to the server log. If it is unchecked, `System.out` output is sent to the system default location only.

## Defining Global Features for Web Applications

You can use the `default-web.xml` file to define features such as filters and security constraints that apply to all web applications.

For example, directory listings are disabled by default for added security. To enable directory listings in your domain's `default-web.xml` file, search for the definition of the servlet whose `servlet-name` is equal to `default`, and set the value of the `init-param` named `listings` to `true`. Then restart the server.

```
<init-param>
  <param-name>listings</param-name>
  <param-value>true</param-value>
</init-param>
```

If `listings` is set to `true`, you can also determine how directory listings are sorted. Set the value of the `init-param` named `sortedBy` to `NAME`, `SIZE`, or `LAST_MODIFIED`. Then restart the server.

```
<init-param>
  <param-name>sortedBy</param-name>
  <param-value>LAST_MODIFIED</param-value>
</init-param>
```

The `mime-mapping` elements in `default-web.xml` are global and inherited by all web applications. You can override these mappings or define your own using `mime-mapping` elements in your web application's `web.xml` file. For more information about `mime-mapping` elements, see the Servlet specification.

You can use the Administration Console to edit the `default-web.xml` file, or edit the file directly using the following steps.

## To Use the `default-web.xml` File

1. Place the JAR file for the filter, security constraint, or other feature in the domain-dir/`lib` directory.
2. Edit the domain-dir/`config/default-web.xml` file to refer to the JAR file.
3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

## Redirecting a URL

You can specify that a request for an old URL be treated as a request for a new URL. This is called redirecting a URL.

To specify a redirected URL for a virtual server, use the `redirect_n` property, where n is a positive integer that allows specification of more than one. Each of these `redirect_n` properties is inherited by all web applications deployed on the virtual server.

The value of each `redirect_n` property has two components which can be specified in any order:

- The first component, `from`, specifies the prefix of the requested URI to match.
- The second component, `url-prefix`, specifies the new URL prefix to return to the client. The `from` prefix is replaced by this URL prefix.

### Example 6-3 Redirecting a URL

This example redirects `from dummy` to `etude`:

```
<property name="redirect_1" value="from=/dummy url-prefix=http://etude"/>
```

## Administering mod\_jk

The Apache Tomcat Connector `mod_jk` can be used to connect the web container with web servers such as Apache HTTP Server. By using `mod_jk`, which comes with Eclipse GlassFish, you can front Eclipse GlassFish with Apache HTTP Server.

You can also use `mod_jk` directly at the JSP/servlet engine for load balancing. For more information about configuring `mod_jk` and Apache HTTP Server for load balancing with Eclipse GlassFish 7 refer to "[Configuring HTTP Load Balancing](#)" in Eclipse GlassFish High Availability Administration Guide.

The following topics are addressed here:

- [To Enable mod\\_jk](#)
- [To Load Balance Using mod\\_jk and Eclipse GlassFish](#)
- [To Enable SSL Between the mod\\_jk Load Balancer and the Browser](#)
- [To Enable SSL Between the mod\\_jk Load Balancer and Eclipse GlassFish](#)

### To Enable mod\_jk

You can front Eclipse GlassFish with Apache HTTP Server by enabling the `mod_jk` protocol for one of Eclipse GlassFish's network listeners, as described in this procedure. A typical use for `mod_jk` would be to have Apache HTTP Server handle requests for static resources, while having requests for dynamic resources, such as servlets and JavaServer Pages (JSPs), forwarded to, and handled by the Eclipse GlassFish back-end instance.

When you use the `jk-enabled` attribute of the network listener, you do not need to copy any additional JAR files into the `/lib` directory. You can also create JK connectors under different virtual servers by using the network listener attribute `jk-enabled`.

1. Install Apache HTTP Server and `mod_jk`.
  - For information on installing Apache HTTP Server, see <http://httpd.apache.org/docs/2.2/install.html>.
  - For information on installing `mod_jk`, see [http://tomcat.apache.org/connector-doc/webserver\\_howto/apache.html](http://tomcat.apache.org/connector-doc/webserver_howto/apache.html).
2. Configure the following files:
  - `apache2/conf/httpd.conf`, the main Apache configuration file
  - `apache2/conf/workers.properties`

[Example 6-4](#) and [Example 6-5](#) provide examples of configuring these two files.

3. Start Apache HTTP Server ([httpd](#)).
4. Start Eclipse GlassFish with at least one web application deployed.

In order for the `mod_jk`-enabled network listener to start listening for requests, the web container must be started. Normally, this is achieved by deploying a web application.

5. Create a jk-enabled network listener by using the [create-network-listener](#) subcommand.

```
asadmin> create-network-listener --protocol http-listener-1 \
--listenerport 8009 --jkenabled true jk-connector
```

6. If you are using the [glassfish-jk.properties](#) file to use non-default values of attributes described at <http://tomcat.apache.org/tomcat-5.5-doc/config/ajp.html>, set the `jk-configuration-file` property of the network listener to the fully-qualified file name of the [glassfish-jk.properties](#) file.

```
asadmin> set server-config.network-config.network-listeners.network-listener.\
jk-connector.jk-configuration-file=domain-dir/config/glassfish-jk.properties
```

7. If you expect to need more than five threads for the listener, increase the maximum threads in the `http-thread-pool` pool:

```
asadmin> set configs.config.server-config.thread-pools.thread-pool.\
http-thread-pool.max-thread-pool-size=value
```

8. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 6-4 `httpd.conf` File for `mod_jk`

This example shows an `httpd.conf` file that is set for `mod_jk`. In this example, `mod_jk` used as a simple pass-through.

```
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 +ForwardURICcompat -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/* worker1
```

#### Example 6-5 `workers.properties` File for `mod_jk`

This example shows a `workers.properties` that is set for `mod_jk`. This `workers.properties` file is referenced in the second line of [Example 6-4](#).

```
# 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
```

#### See Also

For more information on Apache, see <http://httpd.apache.org/>.

For more information on Apache Tomcat Connector, see <http://tomcat.apache.org/connector-doc/index.html>.

## To Load Balance Using `mod_jk` and Eclipse GlassFish

Load balancing is the process of dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time. Load balancing can be configured with or without security.

In order to support stickiness, the Apache `mod_jk` load balancer relies on a `jvmRoute` system property that is included in any `JSESSIONID` received by the load balancer. This means that every Eclipse GlassFish instance that is front-ended by the Apache load balancer must be configured with a unique `jvmRoute` system property.

1. On each of the instances, perform the steps in [To Enable mod\\_jk](#).

If your instances run on the same machine, you must choose different JK ports. The ports must match `worker.worker*.port` in your `workers.properties` file. See the properties file in [Example 6-5](#).

2. On each of the instances, create the `jvmRoute` system property of Eclipse GlassFish by using the `create-jvm-options` subcommand.

Use the following format:

```
asadmin> create-jvm-options "-DjvmRoute=/instance-worker-name"/
```

where `instance-worker-name` is the name of the worker that you defined to represent the instance in the `workers.properties` file.

3. To apply your changes, restart Apache HTTP Server and Eclipse GlassFish.

#### Example 6-6 `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 6-7 `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
```

## To Enable SSL Between the `mod_jk` Load Balancer and the Browser

To activate security for `mod_jk` on Eclipse GlassFish, you must first generate a Secure Socket Layer (SSL) self-signed certificate on the Apache HTTP Server with the `mod_ssl` module. The tasks include generating a private key, a Certificate Signing Request (CSR), a self-signed certificate, and configuring SSL-enabled virtual hosts.

### Before You Begin

The `mod_jk` connector must be enabled.

1. Generate the private key as follows:

```
openssl genrsa -des3 -rand file1:file2:file3:file4:file5 -out server.key 1024
```

where `file1:file2:` and so on represents the random compressed files.

2. Remove the pass-phrase from the key as follows:

```
openssl rsa -in server.key -out server.pem
```

3. Generate the CSR is as follows:

```
openssl req -new -key server.pem -out server.csr
```

Enter the information you are prompted for.

4. Generate a temporary certificate as follows:

```
openssl x509 -req -days 60 -in server.csr -signkey server.pem -out server.crt
```

This temporary certificate is good for 60 days.

5. Create the `http-ssl.conf` file under the `/etc/apache2/conf.d` directory.

6. In the `http-ssl.conf` file, add one of the following redirects:

- Redirect a web application, for example, `JkMount /hello/* worker1`.
- Redirect all requests, for example, `JkMount /* worker1`.

```
# Send all jsp requests to GlassFish
JkMount /*.jsp worker1
```

```
# Send all glassfish-test requests to GlassFish
JkMount /glassfish-test/* loadbalancer
```

#### Example 6-8 `http-ssl.conf` File for `mod_jk` Security

A basic SSL-enabled virtual host will appear in the `http-ssl.conf` file. In this example, all requests are redirected.

```
Listen 443
<VirtualHost _default_:443>
SSLEngine on
SSLCipherSuite ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
SSLCertificateFile "/etc/apache2/2.2/server.crt"
SSLCertificateKeyFile "/etc/apache2/2.2/server.pem"
JkMount /* worker1
</VirtualHost>
```

## To Enable SSL Between the `mod_jk` Load Balancer and Eclipse GlassFish

This procedure does not enable SSL transfer between `mod_jk` and Eclipse GlassFish. It enables `mod_jk` to forward SSL-encrypted information from the browser to Eclipse GlassFish.

### Before You Begin

The self-signed certificate must be configured.

1. Perform the steps in [To Enable mod\\_jk](#).
2. Start another Eclipse GlassFish with at least one web application deployed.

In order for the `mod_jk`-enabled network listener to start listening for requests, the web container must be started. Normally, this is achieved by deploying a web application.

3. Follow instructions from [To Configure an HTTP Listener for SSL](#) on the `mod_jk` connector.

Use the following format:

```
asadmin> create-ssl --type http-listener --certname sampleCert new-listener
```

4. Add the following directives in the `httpd.conf` file under the `/etc/apache2/conf.d` directory:

```
# Should mod_jk send SSL information (default is On)
JkExtractSSL On
# What is the indicator for SSL (default is HTTPS)
JkHTTPSIndicator HTTPS
# What is the indicator for SSL session (default is SSL_SESSION_ID)
```

```
JkSESSIONIndicator SSL_SESSION_ID  
# What is the indicator for client SSL cipher suit (default is SSL_CIPHER )  
JkCIPHERIndicator SSL_CIPHER  
# What is the indicator for the client SSL certificated? (default is  
SSL_CLIENT_CERT )  
JkCERTSIndicator SSL_CLIENT_CERT
```

5. To apply your changes, restart Apache HTTP Server and Eclipse GlassFish.

## Administering mod\_proxy\_ajp

The Apache Connector **mod\_proxy\_ajp** can be used to connect the web container with Apache HTTP Server. By using **mod\_proxy\_ajp**, you can front Eclipse GlassFish with Apache HTTP Server.

### To Enable mod\_proxy\_ajp

You can front Eclipse GlassFish with Apache HTTP Server and its **mod\_proxy\_ajp** connector by enabling the AJP protocol for one of Eclipse GlassFish's network listeners, as described in this procedure. A typical use for **mod\_proxy\_ajp** would be to have Apache HTTP Server handle requests for static resources, while having requests for dynamic resources, such as servlets and JavaServer Pages (JSPs), forwarded to, and handled by the Eclipse GlassFish back-end instance.

1. Install Apache HTTP Server.

For information on installing Apache HTTP Server, see <http://httpd.apache.org/docs/2.2/install.html>.

2. Configure **apache2/conf/httpd.conf**, the main Apache configuration file.

For example:

```
LoadModule proxy_module /usr/lib/httpd/modules/mod_proxy.so  
LoadModule proxy_ajp_module /usr/lib/httpd/modules/mod_proxy_ajp.so  
  
Listen 1979  
NameVirtualHost *:1979  
<VirtualHost *:1979>  
    ServerName localhost  
    ErrorLog /var/log/apache2/ajp.error.log  
    CustomLog /var/log/apache2/ajp.log combined  
  
<Proxy *>  
    AddDefaultCharset Off  
    Order deny,allow  
    Allow from all
```

```
</Proxy>

  ProxyPass / ajp://localhost:8009/
  ProxyPassReverse / ajp://localhost:8009/
</VirtualHost>
```

3. Start Apache HTTP Server ([httpd](#)).
4. Create a jk-enabled network listener by using the `create-network-listener` subcommand.

```
asadmin> create-network-listener --protocol http-listener-1 \
--listenerport 8009 --jkenabled true jk-connector
```

5. If you expect to need more than five threads for the listener, increase the maximum threads in the `http-thread-pool` pool:

```
asadmin> set configs.config.server-config.thread-pools.thread-pool.\
http-thread-pool.max-thread-pool-size=value
```

6. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

## See Also

For more information on Apache, see <http://httpd.apache.org/>.

For more information on the Apache `mod_proxy_ajp` Connector, see [http://httpd.apache.org/docs/2.1/mod/mod\\_proxy.html](http://httpd.apache.org/docs/2.1/mod/mod_proxy.html) and [http://httpd.apache.org/docs/2.1/mod/mod\\_proxy\\_ajp.html](http://httpd.apache.org/docs/2.1/mod/mod_proxy_ajp.html).

For more information on the AJP protocol, see [http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html](http://tomcat.apache.org/-connectors-doc/ajp/ajpv13a.html).

## To Load Balance Using `mod_proxy_ajp` and Eclipse GlassFish

Load balancing is the process of dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time. In the Eclipse GlassFish context, load balancing is most frequently used to distribute work among the instances in a Eclipse GlassFish cluster.

To configure load balancing using `mod_proxy_ajp`, you must use the `mod_proxy_balancer` Apache module in addition to `mod_proxy_ajp`.

In order to support stickiness, the `mod_proxy_balancer` load balancer relies on a `jvmRoute` system property that is included in any `JSESSIONID` received by the load balancer. Consequently, every Eclipse GlassFish instance that is front-ended by the Apache load balancer must be configured with a unique `jvmRoute` system property.

## 1. Install Apache HTTP Server.

For information on installing Apache HTTP Server, see <http://httpd.apache.org/docs/2.2/install.html>.

## 2.

Configure `apache2/conf/httpd.conf`, the main Apache configuration file.

For example:

```
LoadModule proxy_module /usr/lib/httpd/modules/mod_proxy.so
LoadModule proxy_ajp_module /usr/lib/httpd/modules/mod_proxy_ajp.so
LoadModule proxy_balancer_module /usr/lib/httpd/modules/mod_proxy_balancer.so

# Forward proxy needs to be turned off
ProxyRequests Off
# Keep the original Host Header
ProxyPreserveHost On

<Proxy *>
    Order deny,allow
    Deny from all
    Allow from localhost
</Proxy>

# Each BalancerMember corresponds to an instance in the Eclipse GlassFish
# cluster. The port specified for each instance must match the ajp port
# specified for that instance.
<Proxy balancer://localhost>
    BalancerMember ajp://localhost:8009
    BalancerMember ajp://localhost:8010
    BalancerMember ajp://localhost:8011
</Proxy>
```

## 3. Start Apache HTTP Server (`httpd`).

## 4. In Eclipse GlassFish, use the `create-network-listener` subcommand to create a jk-enabled network listener targeted to the cluster.

For example:

```
asadmin> create-network-listener --jkenabled true --target cluster1 \
--protocol http-listener-1 --listenerport ${AJP_PORT} jk-listener
```

In this example, `cluster1` is the name of the cluster and `jk-listener` is the name of the new listener.

## 5. If you expect to need more than five threads for the listener, increase the maximum threads in

the `http-thread-pool` pool:

```
asadmin> set configs.config.cluster1-config.thread-pools.thread-pool.\  
http-thread-pool.max-thread-pool-size=value
```

6. Use the `create-jvm-options` subcommand to create the `jvmRoute` property targeted to the cluster.

For example:

```
asadmin> create-jvm-options --target cluster1 \  
"--DjvmRoute=\${AJP_INSTANCE_NAME}"
```

7. Use the `create-system-properties` subcommand to define the `AJP_PORT` and `AJP_INSTANCE_NAME` properties for each of the instances in the cluster, making sure to match the port values you used in Step 2 when specifying the load balancer members.

For example:

```
asadmin> create-system-properties --target instance1 AJP_PORT=8009  
asadmin> create-system-properties --target instance1 \  
AJP_INSTANCE_NAME=instance1  
asadmin> create-system-properties --target instance2 AJP_PORT=8010  
asadmin> create-system-properties --target instance2 \  
AJP_INSTANCE_NAME=instance2  
asadmin> create-system-properties --target instance3 AJP_PORT=8011  
asadmin> create-system-properties --target instance3 \  
AJP_INSTANCE_NAME=instance3
```

In this example, `instance1`, `instance2` and `instance3` are the names of the Eclipse GlassFish instances in the cluster.

8. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

# 7 Administering the Logging Service

This chapter provides instructions on how to configure logging and how to view log information in the Eclipse GlassFish 7 environment.

The following topics are addressed here:

- [About Logging](#)
- [Default Configuration](#)
- [Configuring the Logging Service](#)
- [Using Asadmin](#)

## About Logging

Logging is the process by which Java Virtual Machine captures information about events that occur, such as important method calls, reaching states, or even configuration errors, security failures, or server malfunction.

This data is recorded in log files and is usually the first source of information when problems occur. Analyzing the log files can help you to follow events that occur in the server runtime and determine the overall health of the server.

Although application components can use other logging frameworks as [SLF4J](#) or [LOG4J2](#), we recommend to use the [Java Util Logging Framework](#) or even better it's latest facade [System.Logger](#).

## Log Manager

Log Manager is a service responsible for the logging system. The service is initialized on JVM startup. After it's first usage it cannot be changed until the JVM is restarted, but it can be reconfigured. Eclipse GlassFish now comes with customized log manager.

## Level

Level is the key feature of the logging system. Every JUL [Logger](#) has an internal integer value representing severity. Levels are set to

- [Log Record](#) as the severity level of the record.
- [Logger](#) as the minimal severity level processed by the logger. Log record with lower severity is ignored.
- [Handler](#) as the minimal severity level processed by the handler. Log record with lower severity is ignored.

There are following predefined levels; however the real usage depends on developers:

- ALL - Special level used by loggers and handlers to declare that they accept all levels.
- SEVERE - Used for serious errors.

- WARNING - Used for log records providing an information about some hazards which can be handled by the application or they can even lead to a severe state.
- INFO - Used to log some important information useful even for the user.
- CONFIG - Used for providing an information related to a configuration.
- FINE - Level for tracing. Used for providing an information about internal behavior of the java application, but still not so detailed.
- FINER - Level for tracing. More detailed information, for example usages of `Logger.entering` and `Logger.exiting` methods.
- FINEST - Level for tracing. Usually very verbose messages slowing down the system but providing a complete information what is going on.
- OFF - Special level used by loggers and handlers to declare that they ignore all levels.

Some projects define custom levels, but at this time it is rather rare.

## Log Record

Log Record is an object created by a `Logger` or its caller and sent to the hierarchy of loggers and handlers which will process it. Every time you use the `Logger` object to log a message with a level passing configured level filters, one `LogRecord` instance is created and processed.

## Logger

`Logger` is a facade of the logging system. It is transparent so it can be initialized as a constant. `Logger` name is usually same as the full name of the class which created it, but specialized loggers can be used too.

Loggers are organized in a tree, so the log record is usually processed by the logger which accepted or created it, then passed to the parent logger (parent package), its parent, etc. unless it is configured to not do so.

The logger log level specifies a severity level to filter what is important for the user. There are several special loggers:

- root logger - uses an empty string as a name.
- system root logger - uses an empty string as a name too, but is not accessible outside JDK, which uses it internally.
- global logger - uses `global` as its name. It is not recommended to use it.

## Handler

`Handler` is responsible for handling the record so it can print the record to the standard output, file, e-mail, network, etc. `Handler` have also its own level set. This level serves as a filter of incoming log records - usually it is not desired to send detailed messages to an e-mail, for example.

## Formatter

Formatter is responsible for formatting of the log record to a String object. It is a usual attribute of the handler, but not all handlers use formatters, for example some handlers may just serialize the log record, call the logging system of the operating system or call a web service.

## Configuration

JUL is usually configured by the `logging.properties` file unless you would use different log manager or you use the JVM option `java.util.logging.config.file` to override it.

# Default Configuration

## The Configuration File

The DAS as well as each configuration, instance, and cluster has its own `logging.properties` file. By default in an Eclipse GlassFish domain, logging properties files are created in the following locations:

Target	Default Location of Logging Properties File
DAS	<code>domain-dir/config/logging.properties</code>
A configuration	<code>domain-dir/config/config-name/logging.properties</code> , where config-name represents the name of a configuration that is shared by one or more instances or clusters.
An instance	<code>domain-dir/config/instance-name-config/logging.properties</code> , where instance-name represents the name of the instance.
A cluster	<code>domain-dir/config/cluster-name-config/logging.properties</code> , where cluster-name represents the name of the cluster.

For information about configuring logging properties, see [Configuring the Logging Service](#).

## The Server Log File

By default Eclipse GlassFish log records are captured in the `server.log` file which can be found in the `logs` directory under the instance's directory. Each instance, managed server instance (that is, each cluster member), and the domain administration server (DAS) has an individual server log file.

This file will contain also logs of deployed applications if they use Java Util Logging, System.Logger or any other facade mapped to this logging system in the backend.

Instance	Default Location
DAS	<code>domain-dir/logs/server.log</code>
Each server instance	<code>instance-dir/logs/server.log</code>
Cluster instance	<code>instance-dir/logs/server.log</code>

For example, in a domain hosted on a given machine that includes a cluster with two managed servers (`ClusterServer1` and `ClusterServer1`) and a standalone instance (`StandaloneServer`), the log files might be arranged in the following directory structure. In this directory structure, the `server.log` file for the DAS is located in `domain-dir/logs`.

```
as-install-parent directory
  glassfish/
    domains/
      domain-dir/
        logs/
          server.log
    nodes/
      hostname/
        ClusterServer1/
          logs/
            server.log
        ClusterServer2/
          logs/
            server.log
      StandaloneServer/
        logs/
          server.log
```

The `server.log` file uses the `ODL` log format by default and is rolled to a new file after it's size exceeds 100 Megabytes. If something in server's JVM prints to the standard output stream or standard error stream, it is redirected to the `server.log` file.

You can change the default name, location, formatting or management of a log file by modifying the `logging properties` file for the corresponding instance, however we don't recommend to change the location of the file as it may affect availability of some services.

## The Access Log File

The `access.log` file serves to log all requests made to the HTTP service or virtual server. This feature is disabled by default, but you can enable it by using the `asadmin set` command, using Admininstration Console or the Admin REST API.

This logging feature is not persisted in `logging.properties` but in `domain.xml`, because it doesn't use Java Util Logging framework but an internal implementation instead.

```
asadmin> get 'server.http-service.*'
server.http-service.virtual-server.__asadmin.access-
log=${com.sun.aas.instanceRoot}/logs/access
server.http-service.virtual-server.__asadmin.access-logging-enabled=inherit
...
server.http-service.virtual-server.server.access-
log=${com.sun.aas.instanceRoot}/logs/access
server.http-service.virtual-server.server.access-logging-enabled=inherit
server.http-service.access-log.buffer-size-bytes=32768
```

```
server.http-service.access-log.format=%client.name% %auth-user-name% %datetime%
%request% %status% %response.length%
server.http-service.access-log.max-history-files=-1
server.http-service.access-log.rotation-enabled=true
server.http-service.access-log.rotation-interval-in-minutes=1440
server.http-service.access-log.rotation-policy=time
server.http-service.access-log.rotation-suffix=yyyy-MM-dd
server.http-service.access-log.write-interval-seconds=300
server.http-service.access-logging-enabled=false
```

## Standard Output Stream

When you start the server with the `--verbose` argument, the server prints log records to the standard output too. The output is limited to just INFO levels and higher and uses the standard error stream, but this can be switched to standard output stream too. Log records are formatted to the [Uniform Log Format](#) by default.

## Logger Levels

The `logging.properties` contains many loggers used by the Eclipse GlassFish to make changes easier. Most of loggers use the INFO level by default.

# Configuring the Logging Service

You can either directly edit the `logging.properties` file or use the `asadmin` command, Administration Console or REST API. On DAS, changes in the file have immediate effect with some small latency before they get applied. For instances managed by nodes it is a bit more complicated and it depends on the synchronization of the configuration with DAS.



If you edit `logging.properties` manually on an instance managed by the node, it will be overwritten on the next synchronization with DAS.

## Loggers

Changing the logger level is quite easy and it is a preferred way how to filter log records by their importance.

So for example if you want to get all records handled by the logging system, you comment out all logger level settings except the root logger and set its level to `FINEST`.

```
.level=FINEST
```

## Handlers

You can use all JUL features, but some of Eclipse GlassFish features depend on some settings like the existence of the configured `GlassFishLogHandler` and its `server.log` file. Also be careful when changing its configuration as it may affect the performance.

## GlassFishLogHandler

The `org.glassfish.main.jul.handler.GlassFishLogHandler` is used to handle persist log records into the `server.log` file. It is optimized for the best performance so logging would not reduce the performance of the server instance and applications deployed to it.

Example:

```
org.glassfish.main.jul.handler.GlassFishLogHandler.buffer.capacity=10000
org.glassfish.main.jul.handler.GlassFishLogHandler.buffer.timeoutInSeconds=0
org.glassfish.main.jul.handler.GlassFishLogHandler.enabled=true
org.glassfish.main.jul.handler.GlassFishLogHandler.encoding=UTF-8
org.glassfish.main.jul.handler.GlassFishLogHandler.file=${com.sun.aas.instanceRoot}/logs/server.log
org.glassfish.main.jul.handler.GlassFishLogHandler.flushFrequency=1
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter=org.glassfish.main.jul.formatter.ODLLogFormatter
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.excludedFields=
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.multiline=true
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.printSource=false
org.glassfish.main.jul.handler.GlassFishLogHandler.level=ALL
org.glassfish.main.jul.handler.GlassFishLogHandler.redirectStandardStreams=true
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.compress=false
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.maxArchiveFiles=0
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.megabytes=100
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.minutes=0
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.rollOnDateChange=false
```

### Basic Settings

- `enabled` - `false` means that the handler will stay configured in the logging system, but it will ignore incoming records. Default is `true`.
- `encoding` - file's character encoding. Default is `UTF-8`.
- `file` - the output file; you can use also system options - default is `${com.sun.aas.instanceRoot}/logs/server.log`
- `flushFrequency` - count of records to be handled in a single batch. Default is 1.
- `formatter` - a formatter class to be used for formatting log records as strings. Default value is `org.glassfish.main.jul.formatter.ODLLogFormatter`
- `formatter.*` - can be used for custom settings of the formatter. This works only for `ODLLogFormatter`, `UniformLogFormatter` and `OneLineFormatter`.
- `level` - Level used to filter log records. Less important log records will be ignored. Default is `ALL`.
- `redirectStandardStreams` - if `true`, which is default, everything printed to the standard output stream and standard error stream is processed by the handler as an INFO resp. ERROR log record. While using these streams is not recommended in Jakarta EE applications, it should be rather rare.

## Receive Buffer

The [GlassFishLogHandler](#) has a receive buffer for incoming log records to optimize throughput. If the buffer is full and a logger tries to add another record, logger's thread is blocked. Then if the timeout is set to 0, the thread is blocked until there's free capacity available - if the handler cannot process records, it may be blocked forever. But if you set the timeout to a positive value, and the thread is blocked for longer time, the whole buffer is reset and added is just an error record describing what happened.

Despite this situation should not occur in standard situation, it may happen ie. when the file system stops working or is extremely slow.

- [buffer.capacity](#) - count of records in the receive buffer, default is 10000 log records.
- [buffer.timeoutInSeconds](#) - maximal time for waiting. Default is 0 which means forever. The buffer is reset after timeout, which means that all unprocessable log records are lost.

## Log File Rotation

The [GlassFishLogHandler](#) can roll the output log file under following conditions:

- if user forced him to do so, see [Forced Rotation of the server.log File](#) for more
- if the size of the file exceeded given limit
- if the date changed
- if the specified number of minutes have passed since the file was opened

The last two conditions are exclusive, the date change has higher priority.

The [flushFrequency](#) parameter affects how many log records will be formatted into the log file before the file is rolled out even after the file size exceeded its configured limit.

The rotation means that the log file is renamed, so the new file name gets a current timestamp as a suffix. If there already is a file with the same name, the implementation tries to add another suffix with a counter until it finds a name which doesn't exist yet.

```
drwxrwxr-x 14 admin admin 4096 čec 29 21:21 ../
-rw-rw-r-- 1 admin admin 2521 srp 3 18:18 server.log
-rw-rw-r-- 1 admin admin 191391 čec 29 21:21 server.log_2022-07-29T21-21-54
-rw-rw-r-- 1 admin admin 24920 srp 3 18:18 server.log_2022-08-03T18-18-38
```

You can configure the logging service to change the default settings for log file rotation, as explained in [Setting Log File Rotation](#).

This is a list of related configuration properties:

- [rotation.compress](#) - compress the rotated file using GZIP algorithm provided by JDK. Default is false.
- [rotation.maxArchiveFiles](#) - maximal count of archived log files (excludes the active one). Default is 0, which means unlimited.

- `rotation.limit.megabytes` - size of the file initiating rotation of the file. Default is 100 Megabytes. The final file will be slightly larger.
- `rotation.limit.minutes` - number of minutes since the last rotation. Default is 0 (unlimited, disabled).
- `rotation.rollOnDateChange` - if set to `true` rolls the file at midnight. Default is `false`.

## SimpleLogHandler

The `org.glassfish.main.jul.handler.SimpleLogHandler` has similar targets as the '`ConsoleHandler`', with few differences:

- can be configured to use `STDOUT` instead of `STDERR`
- uses `OneLineFormatter` by default

The handler configuration properties:

- `encoding` - output character encoding. Default is null which means it will use the system default.
- `formatter` - a formatter class to be used for formatting log records as strings. Default value is `org.glassfish.main.jul.formatter.OneLineFormatter`
- `formatter.*` - can be used for custom settings of the formatter. This works only for `ODLLogFormatter`, `UniformLogFormatter` and `OneLineFormatter`.
- `level` - Level used to filter log records. Less important log records will be ignored. Uses `INFO` as a default.
- `useErrorStream` - if `false`, uses `STDOUT` instead of `STDERR`. Default is `true`.

Configuration example:

```
org.glassfish.main.jul.handler.SimpleLogHandler.formatter=org.glassfish.main.jul.formatter.OneLineFormatter
org.glassfish.main.jul.handler.SimpleLogHandler.formatter.printSource=false
org.glassfish.main.jul.handler.SimpleLogHandler.level=INFO
org.glassfish.main.jul.handler.SimpleLogHandler.useErrorStream=true
```

## SyslogHandler

The `org.glassfish.main.jul.handler.SyslogHandler` is a special handler which is able to send log records to the Unix Syslog facility. The prerequisite is that the Syslog service must listen on the configured network endpoint. See [Syslog on Wikipedia.org](#) for more.

The handler configuration properties:

- `buffer.capacity` - count of records in the receive buffer. Default is 5000 log records.
- `buffer.timeoutInSeconds` - maximal time for waiting. Default is 300. The buffer is reset after timeout, which means that all unprocessable log records are lost.
- `enabled` - `false` means that the handler will stay configured in the logging system, but it will

ignore incoming records. Default is `true`.

- `encoding` - output character encoding used to send data to the Syslog service. Default is `UTF-8`.
- `formatter` - a formatter class to be used for formatting log records as strings. Default value is `java.util.logging.SimpleFormatter`
- `host` - a host name or IP address used of the UDP endpoint. Default is an autodetected name of the local host.
- `level` - Level used to filter log records. Less important log records will be ignored. The default level is `WARNING`.
- `port` - a port of the Syslog UDP listener. Default is 514.

Configuration example:

```
org.glassfish.main.jul.handler.SyslogHandler.enabled=true
org.glassfish.main.jul.handler.SyslogHandler.encoding=UTF-8
org.glassfish.main.jul.handler.SyslogHandler.formatter=java.util.logging.SimpleFormatter
org.glassfish.main.jul.handler.SyslogHandler.host=
org.glassfish.main.jul.handler.SyslogHandler.level=SEVERE
org.glassfish.main.jul.handler.SyslogHandler.port=514
```

## Formatters

### Excluded Fields

Some of formatters support exclusion of some of fields. Currently is possible to exclude following fields:

- `tid` - Thread id and name
- `levelValue` - Integer value of the log level.

### ODLLogFormatter

The `org.glassfish.main.jul.formatter.ODLLogFormatter` logs records in the Oracle Diagnostic Logging Format (ODL).

```
[2022-08-01T19:43:29.952291+02:00] [GlassFish 7.0] [INFO] []
[com.sun.enterprise.server.logging.LogManagerService] [tid: _ThreadID=1
_ThreadName=main] [levelValue: 800] []
Using property file:
/app/appservers/glassfish7/glassfish/domains/domain1/config/logging.properties]

[2022-08-01T19:43:29.986871+02:00] [GlassFish 7.0] [INFO] [NCLS-LOGGING-00009]
[com.sun.enterprise.server.logging.LogManagerService] [tid: _ThreadID=1
_ThreadName=main] [levelValue: 800] []
Running GlassFish Version: Eclipse GlassFish 7.0.0 (build master-b827-g71a6150 2022-
```

```
08-01T11:18:51+0200)]]
```

The formatter has following properties:

- **excludedFields** - comma separated list of fields which should not be printed. None by default.  
See [Excluded Fields](#)
- **fieldSeparator** - String separating fields. Space by default.
- **multiline** - if set to `true` (default), the end of line character is inserted before the log message.
- **printSequenceNumber** - if set to `true`, logs the sequence number of each log record. Default is `false`.
- **printSource** - if set to `true`, logs the class and method which created the log record. Default is `false`.
- **timestampFormat** - see the [DateTimeFormatter](#) documentation. Default is `ISO-8601` timestamp with microseconds and time zone.

## UniformLogFormatter

The `org.glassfish.main.jul.formatter.UniformLogFormatter` logs records in the Uniform Logging Format.

```
[#|2022-08-02T18:16:29.677628+02:00|INFO|GlassFish  
7.0|com.sun.enterprise.server.logging.LogManagerService|_ThreadID=1;_ThreadName=main;_  
LevelValue=800;|  
Using property file:  
/app/appservers/glassfish7/glassfish/domains/domain1/config/logging.properties|#]  
  
[#|2022-08-02T18:16:29.755356+02:00|INFO|GlassFish  
7.0|com.sun.enterprise.server.logging.LogManagerService|_ThreadID=1;_ThreadName=main;_  
LevelValue=800;_MessageID=NCLS-LOGGING-00009;|  
Running GlassFish Version: Eclipse GlassFish 7.0.0 (build master-b827-g71a6150 2022-  
08-01T11:18:51+0200)|#]
```

The formatter has following properties:

- **excludedFields** - comma separated list of fields which should not be printed. None by default.  
See [Excluded Fields](#)
- **fieldSeparator** - String separating fields. Space by default.
- **multiline** - if set to `true` (default), the end of line character is inserted before the log message.
- **printSequenceNumber** - if set to `true`, logs the sequence number of each log record. Default is `false`.
- **printSource** - if set to `true`, logs the class and method which created the log record. Default is `false`.
- **recordMarker.begin** - the prefix of the log record, default is `|#|`.
- **recordMarker.end** - the suffix of the log record, default is `|#|`.

- **timestampFormat** - see the [DateTimeFormatter](#) documentation. Default is [ISO-8601](#) timestamp with microseconds and time zone.

## OneLineFormatter

The [org.glassfish.main.jul.formatter.OneLineFormatter](#) logs records in the following simple format:

```
22:50:43.174228    INFO          main
com.sun.enterprise.server.logging.LogManagerService Using property file:
/app/appservers/glassfish7/glassfish/domains/domain1/config/logging.properties
22:50:43.266648    INFO          main
com.sun.enterprise.server.logging.LogManagerService Running GlassFish Version: Eclipse
GlassFish 7.0.0 (build master-b827-g71a6150 2022-08-01T11:18:51+0200)
```

- **printSource** - if set to `true` (default), logs the class and method which created the log record while when set to `false` it prefers the logger name.
- **size.level** - number of characters taken by the level column. Default is 7.
- **size.thread** - number of characters taken by the thread column. Default is 20.
- **size.class** - number of characters taken by the class name column. Default is 60.
- **timestampFormat** - see the [DateTimeFormatter](#) documentation. Default is [ISO-8601](#) time with microseconds (not date, no timezone).

## SimpleFormatter

The full name is [java.util.logging.SimpleFormatter](#). It is a default formatter provided by the JDK, simple but very flexible. Its most important property is **format**. Read the [documentation of the SimpleFormatter class](#) for more.

# Using Asadmin

Each instance in an Eclipse GlassFish domain has a dedicated `server.log` file, and each instance and cluster has its own `logging.properties` file. To configure logging for an instance or a cluster, Eclipse GlassFish allows you target specific log files or logging properties files when you do the following:

- Set log levels
- Rotate `server.log` files or compress them into a ZIP archive
- Change logging property attributes
- List log levels or log attributes

The following subcommands optionally accept a target specification. A target can be a configuration name, server name, cluster name, or instance name, and is specified as either an operand or as a value passed using the `--target` option. If no target is specified when using any of these subcommands, the default target is the DAS.

Subcommand	Description	Target Specification
<code>collect-log-files</code>	Collects all available log files into a ZIP archive.	--target=target-name
<code>list-log-attributes</code>	Lists logging attributes in the logging properties file.	target-name operand
<code>list-log-levels</code>	Lists the loggers in the logging properties file and their log levels.	target-name operand
<code>rotate-log</code>	Rotates the log file by renaming it and creating a new log file to store new messages.	--target=target-name
<code>set-log-attributes</code>	Sets the specified logging attributes in the logging properties file.	--target=target-name
<code>set-log-file-format</code>	Sets the log file formatter.	--target=target-name
<code>set-log-levels</code>	Sets the log level for one or more loggers listed in the logging properties file.	--target=target-name

This section contains the following examples:

- [To Change the Location of the logging.properties File](#)
- [Setting Log Levels](#)
- [Setting the Log File Format](#)
- [Setting Log File Rotation](#)

## To Change the Location of the logging.properties File

You can set the name and location of the logging properties file by setting the `java.util.logging.config.file` system property.

You have to ensure that the output log file is always used by a single instance. In the default `logging.properties` it is ensured by using the  `${com.sun.aas.instanceRoot}` which always resolves to the instance's root directory. Example:



```
org.glassfish.main.jul.handler.GlassFishLogHandler.file=${com.sun.aas.instanceRoot}/logs/server.log
```

1. Set the `java.util.logging.config.file` system property.

```
asadmin create-jvm-options --target=server-config  
-Djava.util.logging.config.file=/logging.properties
```

Alternatively, you can use the Administration Console to set this system property.

2. To apply your change, restart all instances using this configuration. In our case it would be the DAS:

```
asadmin restart-domain
```

## To Change the Location of the Log File

Even in complex domain you can always find the right `logging.properties` file and update it manually. But probably safer is to use an `asadmin` command to do that.

To change the name and location of the log file, first use the `list-log-attributes` command to obtain the current log attribute setting for the log file name and location. Then use the `set-log-attributes` command to specify the new name or location. The default target for these two commands is the DAS. However, you can optionally specify one of the following targets:

- Configuration name — to target all instances or clusters that share a specific configuration name.
- Server name — to target only a specific server.
- Instance name — to target only a specific instance.
- Cluster name — to target only a specific cluster.

1. Ensure that the DAS is running. Remote commands require a running server.
2. Use the `list-log-attributes` command in remote mode to obtain the current log attribute settings. The name and location of the log file is set with the `org.glassfish.main.jul.handler.GlassFishLogHandler.file` attribute of the `logging properties` file. Optionally you can target a configuration, server, instance, or cluster. If you do not specify a target, the log attribute settings for the DAS are displayed.
3. Use the `set-log-attributes` command in remote mode to define a custom name or location of the log file. If you do not specify a target, the log file for the DAS is targeted by default. If you target a cluster, the name of the cluster log file for each member instance can be changed (the server log file name cannot).

### Example 7-1 Changing the Name and Location of a Cluster's Log File

This example changes the name of the cluster log file for `Cluster1` to `cluster1.log`. `Cluster1` has two server instances: `ClusterServer1` and `ClusterServer2`.

```
asadmin list-log-attributes Cluster1
handlers
<org.glassfish.main.jul.handler.GlassFishLogHandler,org.glassfish.main.jul.handler.Sim
pleLogHandler,org.glassfish.main.jul.handler.SyslogHandler>
org.glassfish.main.jul.handler.GlassFishLogHandler.buffer.capacity      <10000>
org.glassfish.main.jul.handler.GlassFishLogHandler.buffer.timeoutInSeconds   <0>
org.glassfish.main.jul.handler.GlassFishLogHandler.enabled        <true>
org.glassfish.main.jul.handler.GlassFishLogHandler.encoding       <UTF-8>
org.glassfish.main.jul.handler.GlassFishLogHandler.file
```

```
<${com.sun.aas.instanceRoot}/logs/server.log>
org.glassfish.main.jul.handler.GlassFishLogHandler.flushFrequency      <1>
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter
<org.glassfish.main.jul.formatter.ODLLogFormatter>
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.excludedFields   <>
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.multiline <true>
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.printSource
<false>
org.glassfish.main.jul.handler.GlassFishLogHandler.redirectStandardStreams <true>
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.compress <false>
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.megabytes <100>
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.minutes <0>
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.maxArchiveFiles <0>
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.rollOnDateChange
<false>
org.glassfish.main.jul.handler.SimpleLogHandler.formatter
<org.glassfish.main.jul.formatter.UniformLogFormatter>
org.glassfish.main.jul.handler.SimpleLogHandler.formatter.excludedFields      <>
org.glassfish.main.jul.handler.SimpleLogHandler.formatter.printSource <false>
org.glassfish.main.jul.handler.SimpleLogHandler.useErrorStream <true>
org.glassfish.main.jul.handler.SyslogHandler.buffer.capacity <5000>
org.glassfish.main.jul.handler.SyslogHandler.buffer.timeoutInSeconds <300>
org.glassfish.main.jul.handler.SyslogHandler.enabled <false>
org.glassfish.main.jul.handler.SyslogHandler.encoding <UTF-8>
org.glassfish.main.jul.handler.SyslogHandler.formatter
<java.util.logging.SimpleFormatter>
org.glassfish.main.jul.handler.SyslogHandler.host      <>
org.glassfish.main.jul.handler.SyslogHandler.port      <514>
Command list-log-attributes executed successfully.
```

```
asadmin set-log-attributes --target Cluster1
org.glassfish.main.jul.handler.GlassFishLogHandler.file=\${com.sun.aas.instanceRoot}/logs/cluster1.log
```

```
org.glassfish.main.jul.handler.GlassFishLogHandler.file logging attribute value set to
\${com.sun.aas.instanceRoot}/logs/cluster1.log.
The logging attributes are saved successfully for cluster-config.
```

Command set-log-attributes executed successfully.

```
asadmin list-log-attributes ClusterServer1
...
org.glassfish.main.jul.handler.GlassFishLogHandler.file
<\${com.sun.aas.instanceRoot}/logs/cluster1.log>
...
```

```
asadmin list-log-attributes ClusterServer2
...
org.glassfish.main.jul.handler.GlassFishLogHandler.file
<\${com.sun.aas.instanceRoot}/logs/cluster1.log>
```

...

## See Also

You can view the full syntax and options of these subcommands by typing `asadmin help list-log-attributes` and `asadmin help set-log-attributes` at the command line.

## Setting Log Levels

The log level determines the granularity of the message as it is described in the chapter [Level](#).

When setting log levels, you can target a configuration, server, instance, or cluster.

Setting log levels is done by using the `set-log-levels` subcommand. Listing log levels is done by using the `list-log-levels` subcommand.

The following topics are addressed here:

- [To List Logger Levels](#)
- [To Set the Logger Log Level](#)
- [To Set the Handler Log Level](#)

### To List Logger Levels

Eclipse GlassFish provides the means to list all loggers and their log levels. Listing the loggers provides a convenient means to view current loggers and log levels either prior to or after making log level changes.

Use the `list-log-levels` subcommand in remote mode to list the modules and their current log levels. The default target for this subcommand is the DAS. However, you can optionally specify one of the following targets:

- Configuration name — to target all instances or clusters that share a specific configuration name.
- Server name — to target a specific server.
- Instance name — to target a specific instance.
- Cluster name — to target a specific cluster.
  1. Ensure that the DAS is running. Remote subcommands require a running server.
  2. List the existing module loggers and log levels by using the `list-log-levels` subcommand.

### Example 7-2 Listing Logger Levels for DAS

This example shows a partial list of the existing loggers and their log levels in the DAS.

```
asadmin list-log-levels
MBeans <INFO>
com.sun.enterprise.glassfish.bootstrap <INFO>
```

```
com.sun.enterprise.glassfish    <INFO>
com.sun.enterprise.security    <INFO>
com.sun.webui    <INFO>
jakarta.enterprise.admin.rest.client    <INFO>
jakarta.enterprise.admin.connector <INFO>
jakarta.enterprise.admin.rest    <INFO>
jakarta.enterprise.bootstrap    <INFO>
jakarta.enterprise.cluster.gms.admin    <INFO>
jakarta.enterprise.cluster.gms.bootstrap    <INFO>
jakarta.enterprise.cluster.gms    <INFO>
jakarta.enterprise.concurrent    <INFO>
jakarta.enterprise.config.api    <INFO>
...
Command list-log-levels executed successfully.
```

### Example 7-3 Listing Logger Levels for an Instance

This example shows a partial list of the loggers and log levels for the instance [MyServer2](#).

```
asadmin list-log-levels MyServer2
MBeans <INFO>
com.sun.enterprise.glassfish.bootstrap <INFO>
com.sun.enterprise.glassfish    <INFO>
com.sun.enterprise.security    <INFO>
com.sun.webui    <INFO>
cz.acme.level    <ALL>
jakarta.enterprise.admin.rest.client    <INFO>
jakarta.enterprise.admin.connector <INFO>
jakarta.enterprise.admin.rest    <INFO>
jakarta.enterprise.bootstrap    <INFO>
jakarta.enterprise.cluster.gms.admin    <INFO>
jakarta.enterprise.cluster.gms.bootstrap    <INFO>
jakarta.enterprise.cluster.gms    <INFO>
jakarta.enterprise.concurrent    <INFO>
jakarta.enterprise.config.api    <INFO>
...
Command list-log-levels executed successfully.
```

### See Also

You can view the full syntax and options of the subcommand by typing [asadmin help list-log-levels](#) at the command line.

### To Set the Logger Log Level

You will probably need to set logger levels most often. Let's imagine that you would need to set the most verbose logging of an application using the [org.acme](#) package (and logger names).

Then you can edit the [logging.properties](#) file directly, what can be quite more complicated if you

use more than one instance, see the [warning](#).

Safer is to use the `set-log-levels` subcommand:

#### Example 7-5 Changing the Logger Log Level for a Cluster

```
asadmin set-log-levels --target Cluster1 org.acme=ALL  
org.acme package set with log level ALL.These logging levels are set for Cluster1.  
Command set-log-levels executed successfully.
```

#### Example 7-5 Setting Log Levels for Multiple Loggers

The following example sets the log level for security and web container loggers in the DAS.

```
asadmin set-log-levels jakarta.enterprise.system.core.security=FINE\  
:jakarta.enterprise.system.container.web=WARNING  
jakarta.enterprise.system.core.security package set with log level  
FINE.jakarta.enterprise.system.container.web package set with log level WARNING.These  
logging levels are set for server.  
Command set-log-levels executed successfully.
```

#### See Also

You can view the full syntax and options of the subcommand by typing `asadmin help set-log-levels` at the command line.

### To Set the Handler Log Level

The handler log level specifies a severity level filter to prevent overloading of the handler. Default value is usually given by handler's implementation and reflect targets and expected throughput of the handler. For example, you would not want to send all `FINEST` LogRecords by e-mail, but you would like to see them in a local log file.

Because JUL uses the same property syntax for `Logger` levels as for `Handler` levels you can use both `set-log-levels` and `set-log-attributes` subcommands to get the same result (with a bit different syntax).

Both commands in remote mode. The default target for this subcommand is the DAS. However, you can optionally specify one of the following targets using the `--target` option:

- Configuration name — to target all instances or clusters that share a specific configuration name.
  - Server name — to target a specific server.
  - Instance name — to target a specific instance.
  - Cluster name — to target a specific cluster.
1. Ensure that the DAS is running.

2. Set the log level by using the `set-log-attributes` subcommand, specifying the log level of the `org.glassfish.main.jul.handler.GlassFishLogHandler` handler. For example:

```
org.glassfish.main.jul.handler.GlassFishLogHandler <ALL>
```

#### Example 7-6 Changing the Handler Log Level

This example sets the log level for `GlassFishLogHandler` in the DAS to `INFO`:

```
asadmin set-log-attributes  
org.glassfish.main.jul.handler.GlassFishLogHandler.level=INFO  
  
org.glassfish.main.jul.handler.GlassFishLogHandler.level logging attribute value set  
to INFO.  
The logging attributes are saved successfully for server.  
  
Command set-log-attributes executed successfully.
```

#### See Also

You can view the full syntax and options of the subcommand by typing `asadmin help set-log-attributes` at the command line.

## Setting the Log File Format

You can set the format for log records in log files. The following topics are addressed here:

- [To Set the Log File Format](#)
- [To Exclude Fields in Logs](#)
- [To Disable Multiline Mode](#)

### To Set the Log File Format

Use the `set-log-file-format` subcommand in remote mode to set the formatter used by Eclipse GlassFish to format log records in log files. This command is limited to the `GlassFishLogHandler` settings. You can also use the `set-log-attributes` subcommand which is more flexible. Log formats for all server instances in a cluster will be the same. For information about log formats, see [Formatters](#).



Changing the log format forces log rotation to avoid mixed format in the same file.

1. Ensure that the DAS is running. Remote commands require a running server.
2. Set the formatter by using the `set-log-file-format` subcommand.
3. To apply your change, restart affected instances or clusters with the synchronization enabled.

#### Example 7-7 Setting the Log File Format using `set-log-file-format`

This example sets the log file format to `OneLineFormatter` for standalone instance `ManagedServer1` using the `set-log-file-format` subcommand.

```
asadmin set-log-file-format --target ManagedServer1  
org.glassfish.main.jul.formatter.OneLineFormatter  
The log file formatter is set to org.glassfish.main.jul.formatter.OneLineFormatter for  
instance server.  
Command set-log-file-format executed successfully.
```

#### Example 7-8 Setting the Log File Format using `set-log-attributes`

This example sets the log file format to `ULF` for standalone instance `ManagedServer1` using the `set-log-attributes` subcommand.

```
asadmin set-log-attributes --target ManagedServer1 \  
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter=org.glassfish.main.jul.  
formatter.OneLineFormatter  
  
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter logging attribute value  
set to org.glassfish.main.jul.formatter.OneLineFormatter.  
The logging attributes are saved successfully for ManagedServer1-config.  
  
Command set-log-attributes executed successfully.
```

#### See Also

You can view the full syntax and options of the `set-log-file-format` subcommand by typing `asadmin help set-log-file-format` at the command line. You can view the full syntax and options of the `set-log-attributes` subcommand by typing `asadmin help set-log-attributes` at the command line.

#### To Exclude Fields in Logs

Use the `set-log-attributes` subcommand in remote mode to exclude specific name-value fields from log records. If the `excludeFields` attribute is not specified, all name-value fields are included. The following fields can be excluded:

- `tid`
- `levelVal`

1. Ensure that the DAS is running. Remote commands require a running server.
2. Exclude fields by using the `set-log-attributes` subcommand, specifying the attribute and the fields to exclude.
3. To apply your change, restart Eclipse GlassFish.

#### Example 7-9 Excluding Fields in the ODLLogFormatter

This example excludes the `tid` (thread ID and name) and `levelValue` (numerical value of the `Level`)

name-value fields in log records for standalone instance **ManagedServer1**:

```
asadmin set-log-attributes --target ManagedServer1 \
org.glassfish.main.jul.formatter.ODLLogFormatter.excludedFields=tid,levelValue

org.glassfish.main.jul.formatter.ODLLogFormatter.excludedFields logging attribute
value set to tid,levelValue.
The logging attributes are saved successfully for ManagedServer1-config.

Command set-log-attributes executed successfully.
```



If there's the same attribute of the handler's **formatter** property, it has higher priority.

#### Example 7-10 Excluding Fields in the GlassFishLogHandler

This example excludes the **tid** (thread ID and name) and **levelValue** (numerical value of the **Level**) name-value fields in log records for standalone instance **ManagedServer1**:

```
asadmin set-log-attributes --target ManagedServer1 \
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.excludedFields=tid,levelV
alue

org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.excludedFields logging
attribute value set to tid,levelValue.
The logging attributes are saved successfully for ManagedServer1-config.

Command set-log-attributes executed successfully.
```

#### See Also

You can view the full syntax and options of the subcommand by typing **asadmin help set-log-attributes** at the command line.

#### To Disable Multiline Mode

Use the **set-log-attributes** command in remote mode to disable the multiline mode. When multiline mode is enabled (the default), the body of a log message starts on a new line after the message header and is indented.

1. Ensure that the DAS is running. Remote commands require a running server.
2. Set multiline mode by using the **set-log-attributes** subcommand, specifying the formatter attribute and its value (**true** or **false**):
3. To apply your change, restart the instance.

#### Example 7-11 Disabling the Multiline Mode in the log file

Multiline mode is enabled by default. The following example disables multiline mode in log files for standalone instance **ManagedServer1**:

```
asadmin set-log-attributes --target ManagedServer1 \
org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.multiline=false

org.glassfish.main.jul.handler.GlassFishLogHandler.formatter.multiline logging
attribute value set to false.
The logging attributes are saved successfully for ManagedServer1-config.

Command set-log-attributes executed successfully.
```

## See Also

You can view the full syntax and options of the subcommand by typing **asadmin help set-log-attributes** at the command line.

## Setting Log File Rotation

As explained in [The Server Log File](#), Eclipse GlassFish by default rotates the **server.log** file when its size exceeds 100 MB. However, you can change the default rotation settings. For example, you can change the file size limit at which the server rotates the log file or you can configure a server to rotate log files based on a time interval. In addition to changing when rotation occurs, you can also:

- Specify the maximum number of rotated files that can accumulate. By default, Eclipse GlassFish does not limit the number of rotated log files that are retained. However, you can set a limit. After the number of log files reaches this limit, subsequent file rotations delete the oldest rotated log file.
- Rotate the log file manually. A manual rotation forces the immediate rotation of the target log file.

Changing the default log rotation settings is done using the **set-log-attributes** subcommand, and rotating log files manually is done using the **rotate-log** subcommand, as explained in the following sections:

- [To Change the Rotation File Size](#)
- [To Change the File Rotation Interval](#)
- [To Change the Limit Number of Archive Log Files](#)
- [To Rotate Log Files Manually](#)

### To Change the Rotation File Size

Use the **set-log-attributes** subcommand in remote mode to change the log rotation file size. The default target of this subcommand is the DAS. Optionally, you can target a configuration, server, instance, or cluster.

1. Ensure that the DAS is running.

2. Change the rotation file size limit by using the `set-log-attributes` subcommand, specifying the attribute and the desired limit in megabytes:

```
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.megabytes=1000
```

3. Changes will be applied automatically after saving the change to the instance's `logging.properties` file.

#### Example 7-12 Changing the Rotation Size

The following example sets the log file rotation size to 1 MB for the standalone instance `ManagedServer1`:

```
asadmin set-log-attributes --target ManagedServer1 \
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.megabytes=1000

org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.megabytes logging
attribute value set to 1000.
The logging attributes are saved successfully for ManagedServer1-config.

Command set-log-attributes executed successfully.
```

#### See Also

You can view the full syntax and options of the subcommand by typing `asadmin help set-log-attributes` at the command line.

### To Change the File Rotation Interval

Use the `set-log-attributes` subcommand in remote mode to change the log file rotation time limit interval. The default target of this subcommand is the DAS. Optionally, you can target a configuration, server, instance, or cluster. The default value is `0`.

1. Ensure that the DAS is running.
2. Change the rotation time limit by using the `set-log-attributes` subcommand, specifying the following attribute and the desired limit in minutes:

```
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.minutes=minutes
```

3. Changes will be applied automatically after saving the change to the instance's `logging.properties` file.

#### Example 7-13 Changing the Rotation Interval

The following example sets the log file rotation time limit for the cluster `Cluster1`, and all its instances.

```
asadmin set-log-attributes --target Cluster1 \
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.minutes=60

org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.limit.minutes logging
attribute value set to 60.
The logging attributes are saved successfully for cluster-config.

Command set-log-attributes executed successfully.
```

## See Also

You can view the full syntax and options of the subcommand by typing `asadmin help set-log-attributes` at the command line.

## To Change the Limit Number of Archive Log Files

Use the `set-log-attributes` subcommand in remote mode to change the limit on the number of log files that the server creates to store old log messages. The default target of this subcommand is the DAS. Optionally, you can target a configuration, server, instance, or cluster. The default limit value is `0`, which results in no limit placed on the number of rotated log files that are retained.

1. Ensure that the DAS is running.
2. Change the limit number of retained log files by using the `set-log-attributes` subcommand, specifying the following attribute and the desired file limit number:

```
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.maxArchiveFiles=number
```

3. Changes will be applied automatically after saving the change to the instance's effective `logging.properties` file.

### Example 7-14 Changing the Limit Number of Archived Log Files

The following example sets the log limit number of retained log files for the DAS to `10`.

```
asadmin set-log-attributes \
org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.maxArchiveFiles=10

org.glassfish.main.jul.handler.GlassFishLogHandler.rotation.maxArchiveFiles logging
attribute value set to 10.
The logging attributes are saved successfully for server.

Command set-log-attributes executed successfully.
```

## See Also

You can view the full syntax and options of the subcommand by typing `asadmin help set-log-attributes` at the command line.

## To Rotate Log Files Manually

You can rotate log files manually by using the `rotate-log` subcommand in remote mode. The default target of this subcommand is the DAS. Optionally, you can target a configuration, server, instance, or cluster. When you use this subcommand, the target log file is immediately moved to a new time-stamped file and a new log file is created.

Because log rotation is a dynamic operation, you do not need to restart Eclipse GlassFish for changes to take effect.

1. Ensure that the target server or cluster is running.
2. Rotate log files by using the `rotate-log` subcommand.

### Example 7-15 Rotating Log Files Manually

The following example rotates the `server.log` file for `ManagedServer2` to `server.log_yyyy-mm-dd'T'hh-mm-ss`, where `yyyy-mm-dd'T'hh-mm-ss` represents the time when the file is rotated, and creates a new `server.log` file.

```
asadmin rotate-log --target ManagedServer2
Rotated log on instance named 'ManagedServer2'.
Command rotate-log executed successfully.
```

### See Also

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

## Viewing Log Records

The recommended means for general viewing of logging information is to use the Log Viewer in the Administration Console. The Log Viewer simplifies reading, searching, and filtering log file contents. For instructions, see the Administration Console online help.

Eclipse GlassFish also allows you to collect log files into a ZIP archive, which provides the means to obtain and view log files for an instance or cluster even when it is not currently running. The following section explains how to collect all available log files for an instance or cluster and compile them into a single ZIP archive, which is done by using the `collect-log-files` subcommand.

## To Collect Log Files into a ZIP Archive

Use the `collect-log-files` subcommand in remote mode to collect log files into a ZIP archive. The default target of this subcommand is the DAS. Optionally you can target a configuration, server, instance, or cluster.

1. Ensure that the target server or cluster is running. Remote subcommands require a running server.
2. Use the `collect-log-files` subcommand to create the ZIP archive.

The default location in which the ZIP archive is created is the domain-dir/`collected-logs` directory. The `collect-log-files` subcommand allows you to specify a nondefault directory in which the ZIP archive is to be created by using the `--retrieve` option set to `true`, followed by the directory name.

The name of the ZIP file contains the timestamp, as follows:

`log_YYYY-MM-DD_HH-MIN-SEC.zip`

#### Example 7-16 Collecting and Downloading Log Files as a ZIP File

This example shows collecting the log files for the cluster `Cluster1` and compiling them into a ZIP archive in the `/tmp/space/output` directory.

```
asadmin collect-log-files --target Cluster1 --retrieve true /tmp/space/output
Log files are downloaded for ClusterServer1.
Log files are downloaded for ClusterServer2.
Created Zip file under /tmp/space/output/log_2022-08-06_14-57-53.zip.
Command collect-log-files executed successfully.
```

When the ZIP file created by the preceding command is uncompressed, the following directory structure is created:

```
as-install-parent/
  glassfish/
    domains/
      domain-dir/
        collected_logs/
          logs/
            ClusterServer1/
              server.log
            ClusterServer2/
              server.log
```

#### See Also

You can view the full syntax and options of the subcommand by typing `asadmin help collect-log-files` at the command line.

## Listing Loggers

You can list and view information about all public loggers in your distribution of Eclipse GlassFish.

### To List Loggers

Use the `list-loggers` subcommand in remote mode to list the logger name, subsystem, and description of subsystem loggers in your distribution of Eclipse GlassFish. Class name based loggers are not listed.

1. Ensure that the DAS is running. Remote commands require a running server.
2. List loggers by using the `list-loggers` subcommand.

#### Example 7-17 Listing Loggers

This example lists the logger name, subsystem, and description for each logger. Some lines of output are omitted from this example for readability.

```
asadmin list-loggers
Logger Name          Subsystem      Logger
Description

...
jakarta.enterprise.system.core      CORE          Core Kernel
jakarta.enterprise.system.core.ee   AS-CORE       Jakarta EE Core
Kernel
jakarta.enterprise.system.core.security SECURITY      Core Security
jakarta.enterprise.system.core.security.web SECURITY    Core-ee Security
Logger
jakarta.enterprise.system.jmx      JMX           JMX System Logger
jakarta.enterprise.system.security.ssl SECURITY - SSL Security - SSL
...
Command list-loggers executed successfully.
```

#### See Also

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

As an alternative you can take a look into the `default-logging.properties` file which should contain all useful basic loggers set to a default level. The same configuration is distributed in the new domain1's `logging.properties` file, so you can print all actual logger names and levels as we described in [To List Logger Levels](#).

# 8 Administering the Monitoring Service

This chapter explains how to monitor the Eclipse GlassFish 7 components and services by using the [asadmin](#) command-line utility. Instructions for configuring JConsole to monitor Eclipse GlassFish resources are also provided.

The following topics are addressed here:

- [About Monitoring](#)
- [Configuring Monitoring](#)
- [Viewing Common Monitoring Data](#)
- [Viewing Comprehensive Monitoring Data](#)
- [Configuring JConsole to View Eclipse GlassFish Monitoring Data](#)

Instructions for monitoring by using the Administration Console are contained in the Administration Console online help.

For information on using REST interfaces for monitoring, see [Using REST Interfaces to Administer Eclipse GlassFish](#).

## About Monitoring

Monitoring is the process of reviewing the statistics of a system to improve performance or solve problems. The monitoring service can track and display operational statistics, such as the number of requests per second, the average response time, and the throughput. By monitoring the state of various components and services deployed in Eclipse GlassFish, you can identify performance bottlenecks, predict failures, perform root cause analysis, and ensure that everything is functioning as expected. Data gathered by monitoring can also be useful in performance tuning and capacity planning.

For this release of Eclipse GlassFish, monitoring is exposed in a modular way so that many client modules can access and display the monitoring statistics. These clients include the Administration Console, the [asadmin](#) utility, AMX, and REST interfaces.

The following topics are addressed here:

- [How the Monitoring Tree Structure Works](#)
- [About Monitoring for Add-on Components](#)
- [Tools for Monitoring Eclipse GlassFish](#)

## How the Monitoring Tree Structure Works

A monitorable object is a component, subcomponent, or service that can be monitored. Eclipse GlassFish uses a tree structure to track monitorable objects. Because the tree is dynamic, the tree changes as Eclipse GlassFish components are added or removed.

In the tree, a monitorable object can have child objects (nodes) that represent exactly what can be monitored for that object. All child objects are addressed using the dot (.) character as a separator. These constructed names are referred to as dotted names. Detailed information on dotted names is available in the [dotted-names\(5ASC\)](#) help page.

The following command lists the monitorable child objects of the instance `server`:

```
asadmin> list --monitor "server.*"

server.applications
server.connector-service
server.http-service
server.jms-service
server.jvm
server.network
server.orb
server.resources
server.security
server.thread-pool
server.transaction-service
server.web
```

Each object is represented by a dotted name. Dotted names can also address specific attributes in monitorable objects. For example, the `jvm` object has a `memory` attribute with a statistic called `maxheapsize`. The following dotted name addresses the attribute:

```
server.jvm.memory.maxheapsize
```

Although an object is monitorable, it is not necessarily being actively monitored. For instructions on activating monitoring, see [Configuring Monitoring](#).

## Tree Structure of Monitorable Objects

Each monitorable object has a hierarchical tree structure. In the tree, a replaceable such as `*statistics` represents the name of the attribute that you can show statistics for.

The following node tree hierarchies are addressed here:

- [Applications Tree Hierarchy](#)
- [Connector Service Tree Hierarchy](#)

- [HTTP Service Tree Hierarchy](#)
- [JMS/Container Service Tree Hierarchy](#)
- [JVM Tree Hierarchy](#)
- [Network Tree Hierarchy](#)
- [ORB Tree Hierarchy](#)
- [Resources Tree Hierarchy](#)
- [Security Tree Hierarchy](#)
- [Thread Pool Tree Hierarchy](#)
- [Transactions Service Tree Hierarchy](#)
- [Web Tree Hierarchy](#)

## Applications Tree Hierarchy

The **applications** tree contains the following nodes:

```
server.applications
|--- application1
|   |--- ejb-module-1
|   |   |--- ejb1 *
|   |   |   |--- bean-cache (for entity/sfsb) *
|   |   |   |--- bean-pool (for slsb/mdb/entity) *
|   |   |   |--- bean-methods
|   |   |   |   |---method1 *
|   |   |   |   |---method2 *
|   |   |   |--- timers (for s1sb/entity/mdb) *
|--- web-module-1
|   |--- virtual-server-1 *
|   |   |---servlet1 *
|   |   |---servlet2 *
|--- standalone-web-module-1
|   |   |----- virtual-server-2 *
|   |   |       |---servlet3 *
|   |   |       |---servlet4 *
|   |   |----- virtual-server-3 *
|   |   |       |---servlet3 *(same servlet on different vs)
|   |   |       |---servlet5 *
|--- standalone-ejb-module-1
|   |--- ejb2 *
|   |   |--- bean-cache (for entity/sfsb) *
|   |   |--- bean-pool (for slsb/mdb/entity) *
|   |   |--- bean-methods
|   |   |   |--- method1 *
|   |   |   |--- method2 *
|   |   |--- timers (for s1sb/entity/mdb) *
```

```
|--- jersey-application-1  
|   |--- jersey  
|   |   |--- resources  
|   |   |   |--- resource-0  
|   |   |   |   hitcount  
|   |   |   |   *statistic  
|--- application2
```

An example dotted name might be:

```
server.applications.hello.server.request.maxtime
```

An example dotted name under the EJB **method** node might be:

```
server.applications.ejbsfapp1.ejbsfapp1ejbmod1\.jar.SFApp1EJB1
```

An example Jersey dotted name might be:

```
server.applications.helloworld-webapp.jersey.resources.resource-  
0.hitcount.resourcehitcount-count
```

For available statistics, see [EJB Statistics](#), [Jersey Statistics](#), and [Web Statistics](#).

## Connector Service Tree Hierarchy

The **connector-service** tree holds monitorable attributes for pools such as the connector connection pool. The **connector-service** tree contains the following nodes:

```
server.connector-service  
    resource-adapter-1  
        connection-pools  
            pool-1  
        work-management
```

An example dotted name might be `server.connector-service.resource-adapter-1.connection-pools.pool-1`. For available statistics, see [JMS/Connector Service Statistics](#).

## HTTP Service Tree Hierarchy

The **http-service** tree contains the following nodes:

```
server.http-service
```

```
virtual-server
    request
        *statistic
_asadmin
    request
        *statistic
```

An example dotted name under the virutal-server node might be `server.http-service.virtual-server1.request.requestcount`. For available statistics, see [HTTP Service Statistics](#).

## JMS/Container Service Tree Hierarchy

The **jms-service** tree holds monitorable attributes for connection factories (connection pools for resource adapters) and work management (for Message Queue resource adapters). The **jms-service** tree contains the following nodes:

```
server.jms-service
    connection-factories
        connection-factory-1
    work-management
```

An example dotted name under the `connection-factories` node might be `server.jms-service.connection-factories.connection-factory-1` which shows all the statistics for this connection factory. For available statistics, see [JMS/Connector Service Statistics](#).

## JVM Tree Hierarchy

The `jvm` tree contains the following nodes:

```
server.jvm
    class-loading-system
    compilation-system
    garbage-collectors
    memory
    operating-system
    runtime
```

An example dotted name under the `memory` node might be `server.jvm.memory.maxheapsize`. For available statistics, see [JVM Statistics](#).

## Network Tree Hierarchy

The network statistics apply to the network listener, such as `admin-listener`, `http-listener-1`, `http-listener-2`, `https-listener-1`, and `https-listener-2`.

**listener-2.** The `network` tree contains the following nodes:

```
server.network
    type-of-listener
        keep-alive
            *statistic
        file-cache
            *statistic
        thread-pool
            *statistic
        connection-queue
            *statistic
```

An example dotted name under the `network` node might be `server.network.admin-listener.keep-alive.maxrequests-count`. For available statistics, see [Network Statistics](#).

## ORB Tree Hierarchy

The `orb` tree holds monitorable attributes for connection managers. The `orb` tree contains the following nodes:

```
server.orb
    transport
        connectioncache
            inbound
                *statistic
            outbound
                *statistic
```

An example dotted name might be `server.orb.transport.connectioncache.inbound.connectionsidle-count`. For available statistics, see [ORB Statistics \(Connection Manager\)](#).

## Resources Tree Hierarchy

The `resources` tree holds monitorable attributes for pools such as the JDBC connection pool and connector connection pool. The `resources` tree contains the following nodes:

```
server.resources
    connection-pool
        request
            *statistic
```

An example dotted name might be `server.resources.jdbc-connection-pool1.numconnfree.count`. For available statistics, see [Resource Statistics \(Connection Pool\)](#).

## Security Tree Hierarchy

The security tree contains the following nodes:

```
server.security
  ejb
    *statistic
  web
    *statistic
  realm
    *statistic
```

An example dotted name might be `server.security.realm.realmcount-starttime`. For available statistics, see [Security Statistics](#).

## Thread Pool Tree Hierarchy

The `thread-pool` tree holds monitorable attributes for connection managers, and contains the following nodes:

```
server.thread-pool
  orb
    threadpool
      thread-pool-1
        *statistic
```

An example dotted name might be `server.thread-pool.orb.threadpool.thread-pool-1.averagetimeinqueue-current`. For available statistics, see [Thread Pool Statistics](#).

## Transactions Service Tree Hierarchy

The `transaction-service` tree holds monitorable attributes for the transaction subsystem for the purpose of rolling back transactions. The `transaction-service` tree contains the following nodes:

```
server.transaction-service
  statistic
```

An example dotted name might be `server.transaction-service.activeids`. For available statistics, see [Transaction Service Statistics](#).

## Web Tree Hierarchy

The `web` tree contains the following nodes:

```
server.web
    jsp
        *statistic
    servlet
        *statistic
    session
        *statistic
    request
        *statistic
```

An example dotted name for the `servlet` node might be `server.web.servlet.activeservletsloadedcount`. For available statistics, see [Web Module Common Statistics](#).

## About Monitoring for Add-on Components

An add-on component typically generates statistics that Eclipse GlassFish can gather at runtime. Adding monitoring capabilities enables an add-on component to provide statistics to Eclipse GlassFish in the same way as components that are supplied in the Eclipse GlassFish distributions. As a result, you can use the same administrative interfaces to monitor statistics from any installed Eclipse GlassFish component, regardless of the origin of the component.

## Tools for Monitoring Eclipse GlassFish

The following `asadmin` subcommands are provided for monitoring the services and components of Eclipse GlassFish:

- The `enable-monitoring`, `disable-monitoring`, or the `get` and `set` subcommands are used to turn monitoring on or off. For instructions, see [Configuring Monitoring](#).
- The `monitor type` subcommand is used to display basic data for a particular type of monitorable object. For instructions, see [Viewing Common Monitoring Data](#).
- The `list --monitor` subcommand is used to display the objects that can be monitored with the `monitor` subcommand. For guidelines and instructions, see [Guidelines for Using the list and get Subcommands for Monitoring](#).
- The `get` subcommand is used to display comprehensive data, such as the attributes and values for a dotted name. The `get` subcommand used with a wildcard parameter displays all available attributes for any monitorable object. For additional information, see [Guidelines for Using the list and get Subcommands for Monitoring](#).

# Configuring Monitoring

By default, the monitoring service is enabled for Eclipse GlassFish, but monitoring for the individual modules is not. To enable monitoring for a module, you change the monitoring level for that module to LOW or HIGH. You can choose to leave monitoring OFF for objects that do not need to be monitored.

- LOW. Simple statistics, such as create count, byte count, and so on
- HIGH. Simple statistics plus method statistics, such as method count, duration, and so on
- OFF. No monitoring, no impact on performance

The following tasks are addressed here:

- [To Enable Monitoring](#)
- [To Disable Monitoring](#)

## To Enable Monitoring

Use the `enable-monitoring` subcommand to enable the monitoring service itself, or to enable monitoring for individual modules. Monitoring is immediately activated, without restarting Eclipse GlassFish.

You can also use the `set` subcommand to enable monitoring for a module. Using the `set` command is not a dynamic procedure, so you need to restart Eclipse GlassFish for your changes to take effect.

1. Determine which services and components are currently enabled for monitoring.

```
asadmin> get server.monitoring-service.module-monitoring-levels.*
```

This example output shows that the HTTP service is not enabled (OFF for monitoring), but other objects are enabled:

```
configs.config.server-config.monitoring-service.module-monitoring-levels.web-
container=HIGH
    configs.config.server-config.monitoring-service.module-monitoring-
levels.http-service=OFF
        configs.config.server-config.monitoring-service.module-monitoring-
levels.jvm=HIGH
```

2. Enable monitoring by using the `enable-monitoring` subcommand.

Server restart is not required.

Example 8-1 Enabling the Monitoring Service Dynamically

This example enables the monitoring service without affecting monitoring for individual modules.

```
asadmin> enable-monitoring  
Command enable-monitoring executed successfully
```

#### Example 8-2 Enabling Monitoring for Modules Dynamically

This example enables monitoring for the `ejb-container` module.

```
asadmin> enable-monitoring --level ejb-container=HIGH  
Command enable-monitoring executed successfully
```

#### Example 8-3 Enabling Monitoring for Modules by Using the `set` Subcommand

This example enables monitoring for the HTTP service by setting the monitoring level to HIGH (you must restart the server for changes to take effect).

```
asadmin> set server.monitoring-service.module-monitoring-levels.http-service=HIGH  
Command set executed successfully
```

#### See Also

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

## To Disable Monitoring

Use the `disable-monitoring` subcommand to disable the monitoring service itself, or to disable monitoring for individual modules. Monitoring is immediately stopped, without restarting Eclipse GlassFish.

You can also use the `set` subcommand to disable monitoring for a module. Using the `set` command is not a dynamic procedure, so you need to restart Eclipse GlassFish for your changes to take effect.

1. Determine which services and components currently are enabled for monitoring.

```
asadmin get server.monitoring-service.module-monitoring-levels.*
```

This example output shows that monitoring is enabled for `web-container`, `http-service`, and `jvm`:

```
configs.config.server-config.monitoring-service.module-monitoring-levels.web-  
container=HIGH  
configs.config.server-config.monitoring-service.module-monitoring-  
levels.http-service=HIGH
```

```
configs.config.server-config.monitoring-service.module-monitoring-
levels.jvm=HIGH
```

2. Disable monitoring for a service or module by using the [disable-monitoring](#) subcommand.

Server restart is not required.

#### Example 8-4 Disabling the Monitoring Service Dynamically

This example disables the monitoring service without changing the monitoring levels for individual modules.

```
asadmin> disable-monitoring
Command disable-monitoring executed successfully
```

#### Example 8-5 Disabling Monitoring for Modules Dynamically

This example disables monitoring for specific modules. Their monitoring levels are set to OFF.

```
asadmin> disable-monitoring --modules web-container,ejb-container
Command disable-monitoring executed successfully
```

#### Example 8-6 Disabling Monitoring by Using the [set](#) Subcommand

This example disables monitoring for the HTTP service (you must restart the server for changes to take effect).

```
asadmin> set server.monitoring-service.module-monitoring-levels.http-service=OFF
Command set executed successfully
```

#### See Also

You can also view the full syntax and options of the subcommand by typing [asadmin help disable-monitoring](#) at the command line.

## Viewing Common Monitoring Data

Use the [monitor](#) subcommand to display basic data on commonly-monitored objects.

- [To View Common Monitoring Data](#)
- [Common Monitoring Statistics](#)

## To View Common Monitoring Data

Use the `--type` option of the `monitor` subcommand to specify the object for which you want to display data, such as `httplistener`, `jvm`, `webmodule`. If you use the `monitor` subcommand without specifying a type, an error message is displayed.

Output from the subcommand is displayed continuously in a tabular format. The `--interval` option can be used to display output at a particular interval (the default is 30 seconds).

### Before You Begin

A monitorable object must be configured for monitoring before you can display data on the object. See [To Enable Monitoring](#).

1. Determine which type of monitorable object you want to monitor.

Your choices for 5.0 are `jvm`, `httplistener`, and `webmodule`.

2. Request the monitoring data by using the `monitor` subcommand.

### Example 8-7 Viewing Common Monitoring Data

This example requests common data for type `jvm` on instance `server`.

asadmin> monitor --type jvm server						
UpTime(ms)			Heap	and NonHeap	Memory(bytes)	
current	min	max	low	high		count
9437266	8585216	619642880	0	0	93093888	
9467250	8585216	619642880	0	0	93093888	

### See Also

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

## Common Monitoring Statistics

Common monitoring statistics are described in the following sections:

- [HTTP Listener Common Statistics](#)
- [JVM Common Statistics](#)
- [Web Module Common Statistics](#)

## HTTP Listener Common Statistics

The statistics available for the `httpListener` type are shown in the following table.

Table 8-1 HTTP Listener Common Monitoring Statistics

Statistic	Description
<code>ec</code>	Error count. Cumulative value of the error count
<code>mt</code>	Maximum time. Longest response time for a request; not a cumulative value, but the largest response time from among the response times
<code>pt</code>	Processing time. Cumulative value of the times taken to process each request, with processing time being the average of request processing times over request
<code>rc</code>	Request count. Cumulative number of requests processed so far

## JVM Common Statistics

The statistics available for the `jvm` type are shown in the following table.

Table 8-2 JVM Common Monitoring Statistics

Statistic	Description
<code>count</code>	Amount of memory (in bytes) that is guaranteed to be available for use by the JVM machine
<code>high</code>	Retained for compatibility with other releases
<code>low</code>	Retained for compatibility with other releases
<code>max</code>	The maximum amount of memory that can be used for memory management.
<code>min</code>	Initial amount of memory (in bytes) that the JVM machine requests from the operating system for memory management during startup
<code>UpTime</code>	Number of milliseconds that the JVM machine has been running since it was last started

## Web Module Common Statistics

The statistics available for the `webmodule` type are shown in the following table.

Table 8-3 Web Module Common Monitoring Statistics

Statistic	Description
<code>ajlc</code>	Number of active JavaServer Pages (JSP) technology pages that are loaded
<code>asc</code>	Current active sessions
<code>aslc</code>	Number of active servlets that are loaded
<code>ast</code>	Total active sessions
<code>mjlc</code>	Maximum number of JSP pages that are loaded
<code>mslc</code>	Maximum number of servlets that are loaded
<code>rst</code>	Total rejected sessions
<code>st</code>	Total sessions
<code>tjlc</code>	Total number of JSP pages that are loaded
<code>tslc</code>	Total number of servlets that are loaded

## Viewing Comprehensive Monitoring Data

By applying the `list` and `get` subcommands against the tree structure using dotted names, you can display more comprehensive monitoring data, such as a description of each of the statistics and its unit of measurement.

The following topics are addressed here:

- [Guidelines for Using the `list` and `get` Subcommands for Monitoring](#)
- [To View Comprehensive Monitoring Data](#)
- [Comprehensive Monitoring Statistics](#)

### Guidelines for Using the `list` and `get` Subcommands for Monitoring

The underlying assumptions for using the `list` and `get` subcommands with dotted names are:

- A `list` subcommand that specifies a dotted name that is not followed by a wildcard (\*) lists the current node's immediate children. For example, the following subcommand lists all immediate children belonging to the `server` node:

```
list --monitor server
```

- A `list` subcommand that specifies a dotted name followed by a wildcard of the form `.*` lists a hierarchical tree of child nodes from the specified node. For example, the following subcommand lists all children of the `applications` node, their subsequent child nodes, and so on:

```
list --monitor server.applications.*
```

- A `list` subcommand that specifies a dotted name preceded or followed by a wildcard of the form `*dottedname` or `dotted * name` or `dottedname *` lists all nodes and their children that match the regular expression created by the specified matching pattern.
- A `get` subcommand followed by a `.` or `a` gets the set of attributes and their values that belong to the node specified.

For example, the following table explains the output of the `list` and `get` subcommands used with the dotted name for the `resources` node.

Table 8-4 Example Resources Level Dotted Names

Subcommand	Dotted Name	Output
<code>list --monitor</code>	<code>server.resources</code>	List of pool names.
<code>list --monitor</code>	<code>server.resources.`connection-pool1`</code>	No attributes, but a message saying "Use <code>get</code> subcommand with the <code>--monitor</code> option to view this node's attributes and values."
<code>get --monitor</code>	<code>server.resources.`connection-pool1.*`</code>	List of attributes and values corresponding to connection pool attributes.

For detailed information on dotted names, see the [dotted-names\(5ASC\)](#) help page.

## To View Comprehensive Monitoring Data

Although the `monitor` subcommand is useful in many situations, it does not offer the complete list of all monitorable objects. To work with comprehensive data for an object type, use the `list monitor` and the `get monitor` subcommands followed by the dotted name of a monitorable object.

### Before You Begin

A monitorable object must be configured for monitoring before you can display information about the object. See [To Enable Monitoring](#) if needed.

1. List the objects that are enabled for monitoring by using the `list` subcommand.

For example, the following subcommand lists all components and services that have monitoring enabled for instance `server`.

```
asadmin> list --monitor "*"
server.web
```

```
server.connector-service
server.orb
server.jms-serviceserver.jvm
server.applications
server.http-service
server.thread-pools
```

2. Get data for a monitored component or service by using the `get` subcommand.

#### Example 8-8 Viewing Attributes for a Specific Type

This example gets information about all the attributes for object type `jvm` on instance `server`.

```
asadmin> get --monitor server.jvm.*
server.jvm.class-loading-system.loadedclasscount = 3715
server.jvm.class-loading-system.totalloadedclasscount = 3731
server.jvm.class-loading-system.unloadedclasscount = 16
server.jvm.compilation-system.name-current = HotSpot Client Compiler
server.jvm.compilation-system.totalcompilationtime = 769
server.jvm.garbage-collectors.Copy.collectioncount = 285
server.jvm.garbage-collectors.Copy.collectiontime = 980
server.jvm.garbage-collectors.MarkSweepCompact.collectioncount = 2
server.jvm.garbage-collectors.MarkSweepCompact.collectiontime = 383
server.jvm.memory.committedheapsize = 23498752
server.jvm.memory.committednonheapsize = 13598720
server.jvm.memory.initheapsize = 0
server.jvm.memory.initnonheapsize = 8585216
server.jvm.memory.maxheapsize = 66650112
server.jvm.memory.maxnonheapsize = 100663296
server.jvm.memory.objectpendingfinalizationcount = 0
server.jvm.memory.usedheapsize = 19741184
server.jvm.memory.usednonheapsize = 13398352
server.jvm.operating-system.arch-current = x86
server.jvm.operating-system.availableprocessors = 2
server.jvm.operating-system.name-current = Windows XP
server.jvm.operating-system.version-current = 5.1
server.jvm.runtime.classpath-current = glassfish.jar
server.jvm.runtime.inputarguments-current = []
server.jvm.runtime.managementspecversion-current = 1.0
server.jvm.runtime.name-current = 4372@ABBAGANI_WORK
server.jvm.runtime.specname-current = Java Virtual Machine Specification
server.jvm.runtime.specvendor-current = Sun Microsystems Inc.
server.jvm.runtime.specversion-current = 1.0
server.jvm.runtime.uptime = 84813
server.jvm.runtime.vmname-current = Java HotSpot(TM) Client VM
server.jvm.runtime.vmvendor-current = Sun Microsystems Inc.
server.jvm.runtime.vmversion-current = 1.5.0_11-b03
```

#### Example 8-9 Viewing Monitorable Applications

This example lists all the monitorable applications for instance `server`.

```
asadmin> list --monitor server.applications.*  
server.applications.app1  
server.applications.app2  
server.applications.app1.virtual-server1  
server.applications.app2.virtual-server1
```

#### Example 8-10 Viewing Attributes for an Application

This example gets information about all the attributes for application `hello`.

```
asadmin> get --monitor server.applications.hello.*  
server.applications.hello.server.activatedsessionstotal = 0  
server.applications.hello.server.activejspxpsloadedcount = 1  
server.applications.hello.server.activeservletsloadedcount = 1  
server.applications.hello.server.activesessionscurrent = 1  
server.applications.hello.server.activesessionshigh = 1  
server.applications.hello.server.errorcount = 0  
server.applications.hello.server.expiredsessionstotal = 0  
server.applications.hello.server.maxjspxpsloadedcount = 1  
server.applications.hello.server.maxservletsloadedcount = 0  
server.applications.hello.server.maxtime = 0  
server.applications.hello.server.passivatedsessionstotal = 0  
server.applications.hello.server.persistedsessionstotal = 0  
server.applications.hello.server.processingtime = 0.0  
server.applications.hello.server.rejectedsessionstotal = 0  
server.applications.hello.server.requestcount = 0  
server.applications.hello.server.sessionstotal =  
server.applications.hello.server.totaljspxpsloadedcount = 0  
server.applications.hello.server.totalservletsloadedcount = 0
```

#### Example 8-11 Viewing a Specific Attribute

This example gets information about the `jvm` attribute `runtime.vmversion-current` on instance `server`.

```
asadmin> get --monitor server.jvm.runtime.vmversion-current  
server.jvm.runtime.vmversion-current = 10.0-b23
```

## Comprehensive Monitoring Statistics

You can get comprehensive monitoring statistics by forming a dotted name that specifies the statistic you are looking for. For example, the following dotted name will display the cumulative number of requests for the HTTP service on `virtual-server1`:

```
server.http-service.virtual-server1.request.requestcount
```

The tables in the following sections list the statistics that are available for each monitorable object:

- [EJB Statistics](#)
- [HTTP Service Statistics](#)
- [Jersey Statistics](#)
- [JMS/Connector Service Statistics](#)
- [JVM Statistics](#)
- [Network Statistics](#)
- [ORB Statistics \(Connection Manager\)](#)
- [Resource Statistics \(Connection Pool\)](#)
- [Security Statistics](#)
- [Thread Pool Statistics](#)
- [Transaction Service Statistics](#)
- [Web Statistics](#)

## EJB Statistics

EJBs fit into the tree of objects as shown in [Applications Tree Hierarchy](#). Use the following dotted name pattern to get EJB statistics for an application:

```
server.applications.appname.ejbmodulename.ejbname.bean-cache.statistic
```



EJB statistics for an application are available after the application is executed. If the application is deployed but has not yet been executed, all counts will show default values. When the application is undeployed, all its monitoring data is lost.

Statistics available for applications are shown in the following sections:

- [EJB Cache Statistics](#)
- [EJB Container Statistics](#)
- [EJB Method Statistics](#)
- [EJB Pool Statistics](#)
- [Timer Statistics](#)

### EJB Cache Statistics

Use the following dotted name pattern for EJB cache statistics:

```
server.applications.appname.ejbmodulename.bean-cache.ejbname.statistic
```

The statistics available for EJB caches are listed in the following table.

Table 8-5 EJB Cache Monitoring Statistics

Statistic	Data Type	Description
cachemisses	RangeStatistic	The number of times a user request does not find a bean in the cache.
cachehits	RangeStatistic	The number of times a user request found an entry in the cache.
numbeansincache	RangeStatistic	The number of beans in the cache. This is the current size of the cache.
numpassivations	CountStatistic	Number of passivated beans. Applies only to stateful session beans.
numpassivationerrors	CountStatistic	Number of errors during passivation. Applies only to stateful session beans.
numexpiredsessionsremoved	CountStatistic	Number of expired sessions removed by the cleanup thread. Applies only to stateful session beans.
numpassivationsuccess	CountStatistic	Number of times passivation completed successfully. Applies only to stateful session beans.

## EJB Container Statistics

Use the following dotted name pattern for EJB container statistics:

```
server.applications.appname.ejbmodulename.container.ejbname
```

The statistics available for EJB containers are listed in the following table.

Table 8-6 EJB Container Monitoring Statistics

Statistic	Data Type	Description
createcount	CountStatistic	Number of times an EJB's <code>create</code> method is called.
messagecount	CountStatistic	Number of messages received for a message-driven bean.

Statistic	Data Type	Description
methodreadycount	RangeStatistic	Number of stateful or stateless session beans that are in the <code>MethodReady</code> state.
passivecount	RangeStatistic	Number of stateful session beans that are in <code>Passive</code> state.
pooledcount	RangeStatistic	Number of entity beans in pooled state.
readycount	RangeStatistic	Number of entity beans in ready state.
removecount	CountStatistic	Number of times an EJB's <code>remove</code> method is called.

## EJB Method Statistics

Use the following dotted name pattern for EJB method statistics:

```
server.applications.appname.ejbmodulename.bean-methods.ejbname.statistic
```

The statistics available for EJB method invocations are listed in the following table.

Table 8-7 EJB Method Monitoring Statistics

Statistic	Data Type	Description
executiontime	CountStatistic	Time, in milliseconds, spent executing the method for the last successful/unsuccessful attempt to run the operation. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled on the EJB container.
methodstatistic	TimeStatistic	Number of times an operation is called; the total time that is spent during the invocation, and so on.
totalnumerrors	CountStatistic	Number of times the method execution resulted in an exception. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled for the EJB container.
totalnumsuccess	CountStatistic	Number of times the method successfully executed. This is collected for stateless and stateful session beans and entity beans if monitoring enabled is true for EJB container.

## EJB Pool Statistics

Use the following dotted name pattern for EJB pool statistics:

```
server.applications.appname.ejbmodulename.bean-pool.ejbname.statistic
```

The statistics available for EJB pools are listed in the following table.

Table 8-8 EJB Pool Monitoring Statistics

Statistic	Data Type	Description
jmsmaxmessagesload	CountStatistic	The maximum number of messages to load into a JMS session at one time for a message-driven bean to serve. Default is 1. Applies only to pools for message driven beans.
numbeansinpool	RangeStatistic	Number of EJBs in the associated pool, providing information about how the pool is changing.
numthreadswaiting	RangeStatistic	Number of threads waiting for free beans, giving an indication of possible congestion of requests.
totalbeanscreated	CountStatistic	Number of beans created in associated pool since the gathering of data started.
totalbeansdestroyed	CountStatistic	Number of beans destroyed from associated pool since the gathering of data started.

## Timer Statistics

Use the following dotted name pattern for timer statistics:

```
server.applications.appname.ejbmodulename.timers.ejbname.statistic
```

The statistics available for timers are listed in the following table.

Table 8-9 Timer Monitoring Statistics

Statistic	Data Type	Description
numtimerscreated	CountStatistic	Number of timers created in the system.
numtimersdelivered	CountStatistic	Number of timers delivered by the system.
numtimersremoved	CountStatistic	Number of timers removed from the system.

## HTTP Service Statistics

The HTTP service fits into the tree of objects as shown in [HTTP Service Tree Hierarchy](#).

## HTTP Service Virtual Server Statistics

Use the following dotted name pattern for HTTP service virtual server statistics:

```
server.http-service.virtual-server.request.statistic
```

The HTTP service statistics for virtual servers are shown in the following table.

Table 8-10 HTTP Service Virtual Server Monitoring Statistics

Statistic	Data Type	Description
count200	CountStatistic	Number of responses with a status code equal to 200
count2xx	CountStatistic	Number of responses with a status code in the 2xx range
count302	CountStatistic	Number of responses with a status code equal to 302
count304	CountStatistic	Number of responses with a status code equal to 304
count3xx	CountStatistic	Number of responses with a status code equal in the 3xx range
count400	CountStatistic	Number of responses with a status code equal to 400
count401	CountStatistic	Number of responses with a status code equal to 401
count403	CountStatistic	Number of responses with a status code equal to 403
count404	CountStatistic	Number of responses with a status code equal to 404
count4xx	CountStatistic	Number of responses with a status code equal in the 4xx range
count503	CountStatistic	Number of responses with a status code equal to 503
count5xx	CountStatistic	Number of responses with a status code equal in the 5xx range
countother	CountStatistic	Number of responses with a status code outside the 2xx, 3xx, 4xx, and 5xx range
errorcount	CountStatistic	Cumulative value of the error count, with error count representing the number of cases where the response code was greater than or equal to 400
hosts	StringStatistic	The host (alias) names of the virtual server
maxtime	CountStatistic	Longest response time for a request; not a cumulative value, but the largest response time from among the response times
processingtime	CountStatistic	Cumulative value of the times taken to process each request, with processing time being the average of request processing times over the request count
requestcount	CountStatistic	Cumulative number of requests processed so far

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
state	StringStatistic	The state of the virtual server

## Jersey Statistics

Jersey fits into the tree of objects as shown in [Applications Tree Hierarchy](#).

Use the following dotted name pattern for Jersey statistics:

```
server.applications.jersey-application.jersey.resources.resource-0.hitcount.statistic
```

The statistics available for Jersey are shown in the following table.

Table 8-11 Jersey Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
resourcehitcount	CountStatistic	Number of hits on this resource class
rootresourcehitcount	CountStatistic	Number of hits on this root resource class

## JMS/Connector Service Statistics

The JMS/Connector Service fits into the tree of objects as shown in [JMS/Container Service Tree Hierarchy](#).

JMS/Connector Service statistics are shown in the following sections:

- [Connector Connection Pool Statistics \(JMS\)](#)
- [Connector Work Management Statistics \(JMS\)](#)

### Connector Connection Pool Statistics (JMS)

Use the following dotted name pattern for JMS/Connector Service connection pool statistics:

```
server.connector-service.resource-adapter-1.connection-pool.statistic
```

JMS/Connector Service statistics available for the connector connection pools are shown in the following table.



In order to improve system performance, connection pools are initialized lazily;

that is, a pool is not initialized until an application first uses the pool or the pool is explicitly pinged. Monitoring statistics for a connection pool are not available until the pool is initialized.

Table 8-12 Connector Connection Pool Monitoring Statistics (JMS)

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>averageconnwaittime</code>	CountStatistic	Average wait time of connections before they are serviced by the connection pool.
<code>connectionrequestwaittime</code>	RangeStatistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
<code>numconnfailedvalidation</code>	CountStatistic	Total number of connections in the connection pool that failed validation from the start time until the last sample time.
<code>numconnused</code>	RangeStatistic	Total number of connections that are currently being used, as well as information about the maximum number of connections that were used (the high water mark).
<code>numconnfree</code>	RangeStatistic	Total number of free connections in the pool as of the last sampling.
<code>numconntimedout</code>	CountStatistic	Total number of connections in the pool that timed out between the start time and the last sample time.
<code>numconncreated</code>	CountStatistic	Number of physical connections, in milliseconds, that were created since the last reset.
<code>numconndestroyed</code>	CountStatistic	Number of physical connections that were destroyed since the last reset.
<code>numconnacquired</code>	CountStatistic	Number of logical connections acquired from the pool.
<code>numconnreleased</code>	CountStatistic	Number of logical connections released to the pool.
<code>waitqueuelenght</code>	CountStatistic	Number of connection requests in the queue waiting to be serviced.

## Connector Work Management Statistics (JMS)

Use the following dotted name pattern for JMS/Connector Service work management statistics:

```
server.connector-service.resource-adapter-1.work-management.statistic
```

JMS/Connector Service statistics available for connector work management are listed in the following table.

Table 8-13 Connector Work Management Monitoring Statistics (JMS)

Statistic	Data Type	Description
activeworkcount	RangeStatistic	Number of work objects executed by the connector.
completedworkcount	CountStatistic	Number of work objects that were completed.
rejectedworkcount	CountStatistic	Number of work objects rejected by the Eclipse GlassFish.
submittedworkcount	CountStatistic	Number of work objects submitted by a connector module.
waitqueuelength	RangeStatistic	Number of work objects waiting in the queue before executing.
workrequestwaittime	RangeStatistic	Longest and shortest wait of a work object before it gets executed.

## JVM Statistics

The JVM fits into the tree of objects as shown in [JVM Tree Hierarchy](#).

The statistics that are available for the Virtual Machine for Java platform (Java Virtual Machine) or JVM machine are shown in the following sections:

- [JVM Class Loading System Statistics](#)
- [JVM Compilation System Statistics](#)
- [JVM Garbage Collectors Statistics](#)
- [JVM Memory Statistics](#)
- [JVM Operating System Statistics](#)
- [JVM Runtime Statistics](#)

### JVM Class Loading System Statistics

Use the following dotted name pattern for JVM class loading system statistics:

## server.jvm.class-loading-system.statistic

With Java SE, additional monitoring information can be obtained from the JVM. Set the monitoring level to LOW to enable the display of this additional information. Set the monitoring level to HIGH to also view information pertaining to each live thread in the system. More information about the additional monitoring features for Java SE is available in [Monitoring and Management for the Java Platform](#).

The Java SE monitoring tools are discussed at <http://docs.oracle.com/javase/8/docs/technotes/tools/>.

The statistics that are available for class loading in the JVM for Java SE are shown in the following table.

Table 8-14 JVM Monitoring Statistics for Java SE Class Loading

Statistic	Data Type	Description
<code>loadedclasscount</code>	CountStatistic	Number of classes that are currently loaded in the JVM
<code>totalloadedclasscount</code>	CountStatistic	Total number of classes that have been loaded since the JVM began execution
<code>unloadedclasscount</code>	CountStatistic	Number of classes that have been unloaded from the JVM since the JVM began execution

The statistics available for threads in the JVM in Java SE are shown in the following table.

Table 8-15 JVM Monitoring Statistics for Java SE - Threads

Statistic	Data Type	Description
<code>allthreadids</code>	StringStatistic	List of all live thread ids.
<code>currentthreadcpu time</code>	CountStatistic	CPU time for the current thread (in nanoseconds) if CPU time measurement is enabled. If CPU time measurement is disabled, returns -1.
<code>daemonthreadcount</code>	CountStatistic	Current number of live daemon threads.
<code>monitordeadlockedthreads</code>	StringStatistic	List of thread ids that are monitor deadlocked.
<code>peakthreadcount</code>	CountStatistic	Peak live thread count since the JVM started or the peak was reset.
<code>threadcount</code>	CountStatistic	Current number of live daemon and non-daemon threads.

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>totalstartedthreadcount</code>	CountStatistic	Total number of threads created and/or started since the JVM started.

## JVM Compilation System Statistics

Use the following dotted name pattern for JVM compilation system statistics:

```
server.jvm.compilation-system.statistic
```

The statistics that are available for compilation in the JVM for Java SE are shown in the following table.

Table 8-16 JVM Monitoring Statistics for Java SE Compilation

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>name-current</code>	StringStatistic	Name of the current compiler
<code>totalcompilationtime</code>	CountStatistic	Accumulated time (in milliseconds) spent in compilation

## JVM Garbage Collectors Statistics

Use the following dotted name pattern for JVM garbage collectors statistics:

```
server.jvm.garbage-collectors.statistic
```

The statistics that are available for garbage collection in the JVM for Java SE are shown in the following table.

Table 8-17 JVM Monitoring Statistics for Java SE Garbage Collectors

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>collectioncount</code>	CountStatistic	Total number of collections that have occurred
<code>collectiontime</code>	CountStatistic	Accumulated time (in milliseconds) spent in collection

## JVM Memory Statistics

Use the following dotted name pattern for JVM memory statistics:

```
server.jvm.memory.statistic
```

The statistics that are available for memory in the JVM for Java SE are shown in the following table.

Table 8-18 JVM Monitoring Statistics for Java SE Memory

Statistic	Data Type	Description
committedheapsize	CountStatistic	Amount of heap memory (in bytes) that is committed for the JVM to use
committednonheapsize	CountStatistic	Amount of non-heap memory (in bytes) that is committed for the JVM to use
initheapsize	CountStatistic	Size of the heap initially requested by the JVM
initnonheapsize	CountStatistic	Size of the non-heap area initially requested by the JVM
maxheapsize	CountStatistic	Maximum amount of heap memory (in bytes) that can be used for memory management
maxnonheapsize	CountStatistic	Maximum amount of non-heap memory (in bytes) that can be used for memory management
objectpendingfinalizationcount	CountStatistic	Approximate number of objects that are pending finalization
usedheapsize	CountStatistic	Size of the heap currently in use
usednonheapsize	CountStatistic	Size of the non-heap area currently in use

## JVM Operating System Statistics

Use the following dotted name pattern for JVM operating system statistics:

```
server.jvm.operating-system.statistic
```

The statistics that are available for the operating system for the JVM machine in Java SE are shown in the following table.

Table 8-19 JVM Statistics for the Java SE Operating System

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>arch-current</code>	StringStatistic	Operating system architecture
<code>availableprocessors</code>	CountStatistic	Number of processors available to the JVM
<code>name-current</code>	StringStatistic	Operating system name
<code>version-current</code>	StringStatistic	Operating system version

## JVM Runtime Statistics

Use the following dotted name pattern for JVM runtime statistics:

```
server.jvm.runtime.statistic
```

The statistics that are available for the runtime in the JVM runtime for Java SE are shown in the following table.

Table 8-20 JVM Monitoring Statistics for Java SE Runtime

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>classpath-current</code>	StringStatistic	Classpath that is used by the system class loader to search for class files
<code>inputarguments-current</code>	StringStatistic	Input arguments passed to the JVM; not including arguments to the <code>main</code> method
<code>managementspecversion-current</code>	StringStatistic	Management specification version implemented by the JVM
<code>name-current</code>	StringStatistic	Name representing the running JVM
<code>specname-current</code>	StringStatistic	JVM specification name
<code>specvendor-current</code>	StringStatistic	JVM specification vendor
<code>specversion-current</code>	StringStatistic	JVM specification version
<code>uptime</code>	CountStatistic	Uptime of the JVM (in milliseconds)
<code>vmname-current</code>	StringStatistic	JVM implementation name
<code>vmvendor-current</code>	StringStatistic	JVM implementation vendor
<code>vmversion-current</code>	StringStatistic	JVM implementation version

## Network Statistics

Network fits into the tree of objects as shown in [Network Tree Hierarchy](#).

Network statistics are described in the following sections:

- [Network Keep Alive Statistics](#)
- [Network Connection Queue Statistics](#)
- [Network File Cache Statistics](#)
- [Network Thread Pool Statistics](#)

### Network Keep Alive Statistics

Use the following dotted name pattern for network keep alive statistics:

```
server.network.type-of-listener.keep-alive.statistic
```

Statistics available for network keep alive are shown in the following table.

Table 8-21 Network Keep Alive Statistics

Statistic	Data Type	Description
<code>countconnections</code>	CountStatistic	Number of connections in keep-alive mode.
<code>counttimeouts</code>	CountStatistic	Number of keep-alive connections that timed out.
<code>secondstimeouts</code>	CountStatistic	Keep-alive timeout value in seconds.
<code>maxrequests</code>	CountStatistic	Maximum number of requests allowed on a single keep-alive connection.
<code>countflushes</code>	CountStatistic	Number of keep-alive connections that were closed.
<code>counthits</code>	CountStatistic	Number of requests received by connections in keep-alive mode.
<code>countrefusals</code>	CountStatistic	Number of keep-alive connections that were rejected.

### Network Connection Queue Statistics

Use the following dotted name pattern for network connection queue statistics:

```
server.network.type-of-listener.connection-queue.statistic
```

Statistics available for network connection queue are shown in the following table.

Table 8-22 Network Connection Queue Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>countopenconnections</code>	CountStatistic	The number of open/active connections
<code>countoverflows</code>	CountStatistic	Number of times the queue has been too full to accommodate a connection
<code>countqueued</code>	CountStatistic	Number of connections currently in the queue
<code>countqueued15minutesaverage</code>	CountStatistic	Average number of connections queued in the last 15 minutes
<code>countqueued1minuteaverage</code>	CountStatistic	Average number of connections queued in the last 1 minute
<code>countqueued5minutesaverage</code>	CountStatistic	Average number of connections queued in the last 5 minutes
<code>counttotalconnections</code>	CountStatistic	Total number of connections that have been accepted
<code>counttotalqueued</code>	CountStatistic	Total number of connections that have been queued
<code>maxqueued</code>	CountStatistic	Maximum size of the connection queue
<code>peakqueued</code>	CountStatistic	Largest number of connections that were in the queue simultaneously
<code>ticksotalqueued</code>	CountStatistic	(Unsupported) Total number of ticks that connections have spent in the queue

## Network File Cache Statistics

Use the following dotted name pattern for network file cache statistics:

```
server.network.type-of-listener.file-cache.statistic
```

Statistics available for network file cache are shown in the following table.

Table 8-23 Network File Cache Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>contenthits</code>	CountStatistic	Number of hits on cached file content
<code>contentmisses</code>	CountStatistic	Number of misses on cached file content

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>heapsize</code>	CountStatistic	Current cache size in bytes
<code>hits</code>	CountStatistic	Number of cache lookup hits
<code>infohits</code>	CountStatistic	Number of hits on cached file info
<code>infomisses</code>	CountStatistic	Number of misses on cached file info
<code>mappedmemoriesize</code>	CountStatistic	Size of mapped memory used for caching in bytes
<code>maxheapsize</code>	CountStatistic	Maximum heap space used for cache in bytes
<code>maxmappedmemoriesize</code>	CountStatistic	Maximum memory map size used for caching in bytes
<code>misses</code>	CountStatistic	Number of cache lookup misses data type
<code>opencacheentries</code>	CountStatistic	Number of current open cache entries

## Network Thread Pool Statistics

Use the following dotted name pattern for network thread pool statistics:

```
server.network.type-of-listener.thread-pool.statistic
```

Statistics available for network thread pool are shown in the following table.

Table 8-24 Network Thread Pool Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>corethreads</code>	CountStatistic	Core number of threads in the thread pool
<code>currentthreadcount</code>	CountStatistic	Provides the number of request processing threads currently in the listener thread pool
<code>currentthreadsbusy</code>	CountStatistic	Provides the number of request processing threads currently in use in the listener thread pool serving requests
<code>maxthreads</code>	CountStatistic	Maximum number of threads allowed in the thread pool
<code>totalexecutedtasks</code>	CountStatistic	Provides the total number of tasks, which were executed by the thread pool

## ORB Statistics (Connection Manager)

The ORB fits into the tree of objects as shown in [ORB Tree Hierarchy](#).

Use the following dotted name patterns for ORB statistics:

```
server.orb.transport.connectioncache.inbound.statistic  
server.orb.transport.connectioncache.outbound.statistic
```

The statistics available for the connection manager in an ORB are listed in the following table.

Table 8-25 ORB Monitoring Statistics (Connection Manager)

Statistic	Data Type	Description
<code>connectionsidle</code>	CountStatistic	Total number of connections that are idle to the ORB
<code>connectionsinuse</code>	CountStatistic	Total number of connections in use to the ORB
<code>totalconnections</code>	BoundedRangeStatistic	Total number of connections to the ORB

### Resource Statistics (Connection Pool)

By monitoring connection pool resources you can measure performance and capture resource usage at runtime. Connections are expensive and frequently cause performance bottlenecks in applications. It is important to monitor how a connection pool is releasing and creating new connections and how many threads are waiting to retrieve a connection from a particular pool.

The connection pool resources fit into the tree of objects as shown in [Resources Tree Hierarchy](#).

Use the following dotted name pattern for general connection pool statistics:

```
server.resources.pool-name.statistic
```

Use the following dotted name pattern for application-scoped connection pool statistics:

```
server.applications.application-name.resources.pool-name.statistic
```

Use the following dotted name pattern for module-scoped connection pool statistics:

```
server.applications.application-name.module-name.resources.pool-name.statistic
```

The connection pool statistics are shown in the following tables.

 In order to improve system performance, connection pools are initialized lazily; that is, a pool is not initialized until an application first uses the pool or the pool is

explicitly pinged. Monitoring statistics for a connection pool are not available until the pool is initialized.

Table 8-26 General Resource Monitoring Statistics (Connection Pool)

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>averageconnwaittime</code>	CountStatistic	Average wait-time-duration per successful connection request
<code>connrequestwaittime</code>	RangeStatistic	Longest and shortest wait times, in milliseconds, of connection requests since the last sampling. current value indicates the wait time of the last request that was serviced by the pool
<code>numconnacquired</code>	CountStatistic	Number of logical connections acquired from the pool since the last sampling
<code>numconncreated</code>	CountStatistic	Number of physical connections that were created by the pool since the last reset
<code>numconndestroyed</code>	CountStatistic	Number of physical connections that were destroyed since the last reset
<code>numconnfailedvalidation</code>	CountStatistic	Number of connections in the connection pool that failed validation from the start time until the last sampling time
<code>numconnfree</code>	RangeStatistic	Number of free connections in the pool as of the last sampling
<code>numconnnotsuccessfullymatched</code>	CountStatistic	Number of connections rejected during matching
<code>numconnreleased</code>	CountStatistic	Number of connections released back to the pool since the last sampling
<code>numconnsuccessfullymatched</code>	CountStatistic	Number of connections successfully matched
<code>numconntimedout</code>	CountStatistic	Number of connections in the pool that timed out between the start time and the last sampling time
<code>numconnused</code>	RangeStatistic	Number of connections that are currently being used, as well as information about the maximum number of connections that were used (high water mark)

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>frequesqlqueries</code>	StringStatistic	List of the most frequently used SQL queries (Available only when SQL Tracing is enabled)
<code>numpotentialconnleak</code>	CountStatistic	Number of potential connection leaks
<code>numpotentialstatementleak</code>	CountStatistic	Number of potential statement leaks (Available only when Statement Leak Detection is enabled)
<code>numstatementcachehit</code>	CountStatistic	Number of statements that were found in the statement cache (Available only when the Statement Cache is enabled)
<code>numstatementcachemiss</code>	CountStatistic	Number of statements that were not found in the statement cache (Available only when the Statement Cache is enabled)
<code>waitqueuelength</code>	CountStatistic	Number of connection requests in the queue waiting to be serviced

Table 8-27 Application Specific Resource Monitoring Statistics (Connection Pool)

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>numconnacquired</code>	CountStatistic	Number of logical connections acquired from the pool since the last sampling
<code>numconnreleased</code>	CountStatistic	Number of connections released back to the pool since the last sampling
<code>numconnused</code>	RangeStatistic	Number of connections that are currently being used, as well as information about the maximum number of connections that were used (high water mark)

## Security Statistics

Security fits into the tree of objects as shown in [Security Tree Hierarchy](#).

Statistics available for security are shown in the following sections:

- [EJB Security Statistics](#)
- [Web Security Statistics](#)
- [Realm Security Statistics](#)

## EJB Security Statistics

Use the following dotted name pattern for EJB security statistics:

```
server.security.ejb.statistic
```

The statistics available for EJB security are listed in the following table.

Table 8-28 EJB Security Monitoring Statistics

Statistic	Data Type	Description
policyconfigurationcount	CountStatistic	Number of policy configuration
securitymanagercount	CountStatistic	Number of EJB security managers

## Web Security Statistics

Use the following dotted name pattern for web security statistics:

```
server.security.web.statistic
```

The statistics available for web security are listed in the following table.

Table 8-29 Web Security Monitoring Statistics

Statistic	Data Type	Description
websecuritymanagercount	CountStatistic	Number of security managers
webpolicyconfigurationcount	CountStatistic	Number of policy configuration objects

## Realm Security Statistics

Use the following dotted name pattern for realm security statistics:

```
server.security.realm.statistic
```

The statistics available for realm security are listed in the following table.

Table 8-30 Realm Security Monitoring Statistics

Statistic	Data Type	Description
realmcount	CountStatistic	Number of realms

## Thread Pool Statistics

The thread pool fits into the tree of objects as shown in [Thread Pool Tree Hierarchy](#).

The statistics available for thread pools are shown in the following sections:

- [Thread Pool Monitoring Statistics](#)
- [JVM Statistics for Java SE-Thread Information](#)

### Thread Pool Monitoring Statistics

Use the following dotted name pattern for thread pool statistics:

```
server.thread-pool.thread-pool.statistic
```

The statistics available for the thread pool are shown in the following table.

Table 8-31 Thread Pool Monitoring Statistics

Statistic	Data Type	Description
averagetimeinqueue	BoundedRangeStatistic	Average amount of time (in milliseconds) a request waited in the queue before being processed
averageworkcompletiontime	BoundedRangeStatistic	Average amount of time (in milliseconds) taken to complete an assignment
currentbusythreads	CountStatistic	Number of busy threads
currentnumberofthreads	BoundedRangeStatistic	Current number of request processing threads
numberofavailablethreads	CountStatistic	Number of available threads
numberofworkitemsinqueue	BoundedRangeStatistic	Current number of work items waiting in queue

Statistic	Data Type	Description
<code>totalworkitemsadded</code>	CountStatistic	Total number of work items added to the work queue as of last sampling

## JVM Statistics for Java SE - Thread Information

The statistics available for `ThreadInfo` in the JVM in Java SE are shown in the following table.

Table 8-32 JVM Monitoring Statistics for Java SE - Thread Info

Statistic	Data Type	Description
<code>blockedcount</code>	CountStatistic	Total number of times that the thread entered the <code>BLOCKED</code> state.
<code>blockedtime</code>	CountStatistic	Time elapsed (in milliseconds) since the thread entered the <code>BLOCKED</code> state. Returns -1 if thread contention monitoring is disabled.
<code>lockname</code>	StringStatistic	String representation of the monitor lock that the thread is blocked to enter or waiting to be notified through the <code>Object.wait</code> method.
<code>lockownerid</code>	CountStatistic	ID of the thread that holds the monitor lock of an object on which this thread is blocking.
<code>lockownername</code>	StringStatistic	Name of the thread that holds the monitor lock of the object this thread is blocking on.
<code>stacktrace</code>	StringStatistic	Stack trace associated with this thread.
<code>threadid</code>	CountStatistic	ID of the thread.
<code>threadname</code>	StringStatistic	Name of the thread.
<code>threadstate</code>	StringStatistic	State of the thread.
<code>waitedtime</code>	CountStatistic	Elapsed time (in milliseconds) that the thread has been in a <code>WAITING</code> state. Returns -1 if thread contention monitoring is disabled.
<code>waitedcount</code>	CountStatistic	Total number of times the thread was in <code>WAITING</code> or <code>TIMED_WAITING</code> states.

## Transaction Service Statistics

The transaction service allows the client to freeze the transaction subsystem in order to roll back transactions and determine which transactions are in process at the time of the freeze. The

transaction service fits into the tree of objects as shown in [Transactions Service Tree Hierarchy](#).

Use the following dotted name pattern for transaction service statistics:

```
server.transaction-service.statistic
```

The statistics available for the transaction service are shown in the following table.

Table 8-33 Transaction Service Monitoring Statistics

Statistic	Data Type	Description
activecount	CountStatistic	Number of transactions currently active.
activeids	StringStatistic	The ID's of the transactions that are currently active. Every such transaction can be rolled back after freezing the transaction service.
committedcount	CountStatistic	Number of transactions that have been committed.
rolledbackcount	CountStatistic	Number of transactions that have been rolled back.
state	StringStatistic	Indicates whether or not the transaction has been frozen.

## Web Statistics

The web module fits into the tree of objects as shown in [Web Tree Hierarchy](#).

The available web statistics shown in the following sections:

- [Web Module Servlet Statistics](#)
- [Web JSP Statistics](#)
- [Web Request Statistics](#)
- [Web Servlet Statistics](#)
- [Web Session Statistics](#)

### Web Module Servlet Statistics

Use the following dotted name pattern for web module servlet statistics:

```
server.applications.web-module.virtual-server.servlet.statistic  
server.applications.application.web-module.virtual-server.servlet.statistic
```

The available web module servlet statistics are shown in the following table.

Table 8-34 Web Module Servlet Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>errorcount</code>	CountStatistic	Cumulative number of cases where the response code is greater than or equal to 400.
<code>maxtime</code>	CountStatistic	Maximum amount of time the web container waits for requests.
<code>processingtime</code>	CountStatistic	Cumulative value of the amount of time required to process each request. The processing time is the average of request processing times divided by the request count.
<code>requestcount</code>	CountStatistic	The total number of requests processed so far.
<code>servicetime</code>	CountStatistic	Aggregate response time in milliseconds.

## Web JSP Statistics

Use the following dotted name pattern for web JSP statistics:

```
server.applications.web-module.virtual-server.statistic
server.applications.application.web-module.virtual-server.statistic
```

The available web JSP statistics are shown in the following table.

Table 8-35 Web JSP Monitoring Statistics

<b>Statistic</b>	<b>Data Type</b>	<b>Description</b>
<code>jspcount-current</code>	RangeStatistic	Number of active JSP pages
<code>jsppererrorcount</code>	CountStatistic	Total number of errors triggered by JSP page invocations
<code>jspreloadedcount</code>	CountStatistic	Total number of JSP pages that were reloaded
<code>totaljspcount</code>	CountStatistic	Total number of JSP pages ever loaded

## Web Request Statistics

Use the following dotted name pattern for web request statistics:

```
server.applications.web-module.virtual-server.statistic
server.applications.application.web-module.virtual-server.statistic
```

The available web request statistics are shown in the following table.

Table 8-36 Web Request Monitoring Statistics

Statistic	Data Type	Description
<code>errorcount</code>	CountStatistic	Cumulative value of the error count, with error count representing the number of cases where the response code was greater than or equal to 400
<code>maxtime</code>	CountStatistic	Longest response time for a request; not a cumulative value, but the largest response time from among the response times
<code>processingtime</code>	CountStatistic	Average request processing time, in milliseconds
<code>requestcount</code>	CountStatistic	Cumulative number of the requests processed so far

## Web Servlet Statistics

Use the following dotted name pattern for web servlet statistics:

```
server.applications.web-module.virtual-server.statistic  
server.applications.application.web-module.virtual-server.statistic
```

The available web servlet statistics are shown in the following table.

Table 8-37 Web Servlet Monitoring Statistics

Statistic	Data Type	Description
<code>activeservletsloadedcount</code>	RangeStatistic	Number of currently loaded servlets
<code>servletprocessingtimes</code>	CountStatistic	Cumulative servlet processing times , in milliseconds
<code>totalservletsloadedcount</code>	CountStatistic	Cumulative number of servlets that have been loaded into the web module

## Web Session Statistics

Use the following dotted name pattern for web session statistics:

```
server.applications.web-module.virtual-server.statistic  
server.applications.application.web-module.virtual-server.statistic
```

The available web session statistics are shown in the following table.

Table 8-38 Web Session Monitoring Statistics

Statistic	Data Type	Description
<code>activatedsessionstotal</code>	CountStatistic	Total number of activated sessions
<code>activesessionscurrent</code>	RangeStatistic	Number of currently active sessions
<code>activesessionshigh</code>	CountStatistic	Maximum number of concurrently active sessions
<code>expiredsessionstotal</code>	CountStatistic	Total number of expired sessions
<code>passivatedsessionstotal</code>	CountStatistic	Total number of passivated sessions
<code>persistedsessionstotal</code>	CountStatistic	Total number of persisted sessions
<code>rejectedsessionstotal</code>	CountStatistic	Total number of rejected sessions
<code>sessionstotal</code>	CountStatistic	Total number of sessions created

## Configuring JConsole to View Eclipse GlassFish Monitoring Data

Java SE provides tools to connect to an MBean Server and view the MBeans registered with the server. JConsole is one such popular JMX Connector Client and is available as part of the standard Java SE distribution. When you configure JConsole for use with Eclipse GlassFish, Eclipse GlassFish becomes the JMX Connector's server end and JConsole becomes the JMX connector's client end.

### To Connect JConsole to Eclipse GlassFish

Java SE 6 enhanced management and monitoring of the virtual machine by including a Platform MBean Server and by including managed beans (MBeans) to configure the virtual machine.

To view all MBeans, Eclipse GlassFish provides a configuration of the standard JMX connector server called System JMX Connector Server. As part of Eclipse GlassFish startup, an instance of this JMX Connector Server is started. Any compliant JMX connector client can connect to the server using the JMX Connector Server.

By default, Eclipse GlassFish is configured with a non-secure System JMX Connector Server. If this is an issue, the JMX connector can be removed. However, access can be restricted to a specific IP address (for example, the loopback address) by setting `address` to `localhost`.

1. Start the domain.

For instructions, see [To Start a Domain](#).

## 2. Start JConsole using this format: `JDK_HOME/bin/jconsole`

For example:

```
/usr/java/bin/jconsole
```

The JConsole Connect to Agent window is displayed.

### 3. Click the Remote tab and type the host name and port.

Always connect remotely with JConsole, otherwise MBeans will not load automatically.

### 4. Click Connect.

### 5. In the Remote Process text box, specify the JMX Service URL.

For example:

```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

The JMX Service URL is emitted by the server at startup, looking something like this:

```
[#|2009-12-03T10:25:17.737-0800|INFO|glassfish7.0|
x..system.tools.admin.org.glassfish.server|_ThreadID=20;
_ThreadName=Thread-26;|JMXStartupService: Started JMXConnector, JMXService
URL = service:jmx:rmi://localhost:8686/jndi/rmi://localhost:8686/jmxrmi|#]
```

However, in most cases, simply entering `host:port` is fine, such as, `192.168.1.150:8686`. The long Service URL is not needed.



Another host name can be substituted for `localhost`. The default port number (8686) could change if the `jmx-connector` configuration has been modified.

### 6. Click Connect.

In the JConsole window you will see all your MBeans, JVM information, and so on, in various tabs. Most of the useful MBeans are to be found in the `amx` and `java.lang` domains.

## See Also

For more information about JConsole, see <http://docs.oracle.com/javase/8/docs/technotes/guides/management/jconsole.html>.

# 9 Administering Life Cycle Modules

This chapter provides procedures for administering life cycle modules in the Eclipse GlassFish 7 environment.

The following topics are addressed here:

- [About Life Cycle Modules](#)
- [Configuring Life Cycle Modules](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About Life Cycle Modules

Life cycle modules, also known as initialization services, provide a means of running short or long duration Java-based tasks within the Eclipse GlassFish environment. These modules are automatically initiated at server startup and are notified at various phases of the server life cycle. Configured properties for a life cycle module are passed as properties during server initialization.

All life cycle module classes and interfaces are in the `as-install/modules/glassfish-api.jar` file.

A life cycle module listens for and performs its tasks in response to the following Eclipse GlassFish sequence of events:

1. Initialization. The server reads the configuration, initializes built-in subsystems (such as security and logging services), and creates the containers.
2. Startup. The server loads and initializes deployed applications.
3. Ready. The server begins servicing requests.
4. Shutdown. The server shuts down the applications and stops.
5. Termination. The server closes the containers, the built-in subsystems, and the server runtime environment.

These events are defined in the `LifecycleEvent` class. For information on creating life cycle modules, see "Developing Lifecycle Listeners" in Eclipse GlassFish Application Development Guide.



If the `is-failure-fatal` setting is set to true (the default is false), life cycle module failure prevents server initialization or startup, but not shutdown or termination.

## Configuring Life Cycle Modules

The following topics are addressed here:

- [To Create a Life Cycle Module](#)
- [To List Life Cycle Modules](#)
- [To Update a Life Cycle Module](#)
- [To Delete a Life Cycle Module](#)

## To Create a Life Cycle Module

Use the `create-lifecycle-module` subcommand in remote mode to create a life cycle module.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a new life cycle modules by using the `create-lifecycle-module` subcommand.

Information about options and properties for the subcommand are included in this help page.

3. Restart the server for your changes to take effect.

See [To Restart a Domain](#).

### Example 9-1 Creating a Life Cycle Module

This example creates the `customSetup` life cycle module :

```
asadmin> create-lifecycle-module --classname "com.acme.CustomSetup"
--classpath "/export/customSetup" --loadorder 1 --failurefatal=true
--description "this is a sample customSetup"
--property rmi="Server\=acme1\:7070":timeout=30 customSetup
Command create-lifecycle-module executed successfully
```

### See Also

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

## To List Life Cycle Modules

Use the `list-lifecycle-modules` subcommand in remote mode to list the existing life cycle modules.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List life cycle modules by using the `list-lifecycle-modules` subcommand.

### Example 9-2 Listing Life Cycle Modules

This example lists the existing life cycle modules.

```
asadmin> list-lifecycle-modules  
WSTCPConnectorLCModule  
Command list-lifecycle-modules executed successfully
```

## See Also

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

## To Update a Life Cycle Module

Use the `set` subcommand to update an existing life cycle module.

1. List the properties that can be updated for a life cycle module by using the `get` subcommand.

For example (single mode):

```
asadmin get "*" | grep sampleLCM  
applications.application.sampleLCMmodule.availability-enabled=false  
applications.application.sampleLCMmodule.directory-deployed=false  
applications.application.sampleLCMmodule.enabled=true  
applications.application.sampleLCMmodule.name=sampleLCMmodule  
applications.application.sampleLCMmodule.object-type=user  
applications.application.sampleLCMmodule.property.class-  
name=example.lc.SampleModule  
applications.application.sampleLCMmodule.property.classpath=/build/lcm.jar  
applications.application.sampleLCMmodule.property.is-failure-fatal=false  
applications.application.sampleLCMmodule.property.isLifecycle=true
```

2. Update a life cycle module by using the `oset` subcommand.
3. Restart the server for your changes to take effect.

See [To Restart a Domain](#).

### Example 9-3 Updating a Life Cycle Module

This example updates the `classpath` property.

```
sadmin> set applications.application.sampleLCMmodule.  
property.classpath=/build/lcm_new.jarapplications.application.  
sampleLCMmodule.property.classpath=/build/lcm_new.jar  
Command set executed successfully.
```

## See Also

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

## To Delete a Life Cycle Module

Use the `delete-lifecycle-module` subcommand in remote mode to delete a life cycle module.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the current life cycle modules by using the `list-lifecycle-modules` subcommand.
3. Delete a life cycle module by using the `delete-lifecycle-module` subcommand.

### Example 9-4 Deleting a Life Cycle Module

This example deletes the `customSetup` life cycle module.

```
asadmin> delete-lifecycle-module customSetup
Command delete-lifecycle-module executed successfully
```

### See Also

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

# 10 Administering Batch Jobs

This chapter provides procedures for administering batch jobs in the Eclipse GlassFish environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Batch Jobs](#)
- [Viewing Batch Jobs](#)
- [Configuring the Batch Runtime](#)

Instructions for accomplishing these tasks by using the Administration Console are contained in the Administration Console online help.

## About Batch Jobs

Eclipse GlassFish provides a batch runtime for the scheduling and execution of batch jobs. Batch jobs are typically long-running, bulk-oriented tasks that contain a series of steps and can be executed without user interaction. Examples include billing, report generation, data format conversion, and image processing.

Batch applications submit jobs to the batch runtime and provide instructions about how and when to execute the steps. The batch runtime processes the steps as directed by job XML documents packaged with the applications and stores information about jobs in a job repository. In Eclipse GlassFish, the job repository is a database.

For detailed information about batch jobs, batch processing, and the batch processing framework, see [Batch Processing](#) in The Jakarta EE Tutorial. Also see [Java Specification Request 352: Batch Applications for the Java Platform](#) (<http://jcp.org/en/jsr/detail?id=352>). The specification defines the programming model for batch applications and the runtime for scheduling and executing batch jobs.

## Viewing Batch Jobs

You can view detailed information about batch jobs, executions, and steps. Users who log in to the `asadmin` utility or to the Administration Console as administrator are the only users who can view details for all batch jobs submitted by all applications in the Eclipse GlassFish environment.

The following tasks are used to view information about batch jobs:

- [To List Batch Jobs](#)
- [To List Batch Job Executions](#)
- [To List Batch Job Steps](#)

## To List Batch Jobs

Use the `list-batch-jobs` subcommand in remote mode to list batch jobs and job details.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List batch jobs by using the `list-batch-jobs` subcommand.

### Example 10-1 Listing Batch Jobs

This example lists batch jobs for the default server instance, `server`. Use `list-batch-jobs -l` to list additional details.

```
asadmin> list-batch-jobs
JOBNAME INSTANCECOUNT
payroll 9
bonus 6
Command list-batch-jobs executed successfully.
```

### See Also

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

## To List Batch Job Executions

When the batch runtime executes a job, the execution is given a unique execution ID. An execution ID is similar to a process ID. A new execution is created the first time a job is started and every time the existing execution is restarted.

Use the `list-batch-job-executions` subcommand in remote mode to list batch job executions and execution details.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List batch job executions by using the `list-batch-job-executions` subcommand.

### Example 10-2 Listing Batch Job Executions

This example lists batch job executions for the default server instance, `server`, and displays specific details. Use `list-batch-job-executions -l` to list additional details.

```
asadmin> list-batch-job-executions -o=jobname,executionid,batchstatus,exitstatus
JOBNAME EXECUTIONID BATCHSTATUS EXITSTATUS
payroll 9          COMPLETED   COMPLETED
bonus   6          FAILED      FAILED
```

```
Command list-batch-job-executions executed successfully.
```

## See Also

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

## To List Batch Job Steps

A batch job consists of one or more steps. A step is an independent and sequential phase of a batch job.

Use the `list-batch-job-steps` subcommand in remote mode to list steps and step details for a specific batch job execution.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the execution ID of an execution by using the `list-batch-job-executions` subcommand.
3. List steps for a specific batch job execution by using the `list-batch-job-steps` subcommand.

### Example 10-3 Listing Batch Job Steps

This example lists batch job steps and specific step details for a job execution with the execution ID of 7. The target is the default server instance, `server`. Use `list-batch-job-steps -l` to list additional details.

Some lines of output are omitted from this example for readability.

```
asadmin> list-batch-job-steps o=stepname,stepid,batchstatus,stepmetrics 7
STEPNAME  STEPID  BATCHSTATUS  STEPMETRICS
prepare    7        COMPLETED   METRICNAME      VALUE
                           READ_COUNT     8
                           WRITE_COUNT    8
                           PROCESS_SKIP_COUNT 0
process    8        COMPLETED   METRICNAME      VALUE
                           READ_COUNT     8
                           WRITE_COUNT    8
                           PROCESS_SKIP_COUNT 0
...
Command list-batch-job-steps executed successfully.
```

## See Also

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

# Configuring the Batch Runtime

The batch runtime uses a data source and a managed executor service to execute batch jobs. The data source stores information about current and past jobs, and the managed executor service provides threads to jobs. Batch runtime configuration data is stored in the `config` element in `domain.xml`.

Eclipse GlassFish provides a default data source and managed executor service for the execution of batch jobs. For the domain administration server (DAS), the default data source is `jdbc/TimerPool` and the default managed executor service is `concurrent/defaultManagedExecutorService`. If you create a standalone server instance or a standalone cluster, the default data source is `jdbc/_default`. You can configure the batch runtime to use different resources.

For more information about data sources, see [Administering Database Connectivity](#). For more information about managed executor services, see [Configuring Managed Executor Services](#).

The following tasks are used to view and configure the batch runtime:

- [To List the Batch Runtime Configuration](#)
- [To Configure the Batch Runtime](#)

## To List the Batch Runtime Configuration

Use the `list-batch-runtime-configuration` subcommand in remote mode to display the configuration of the batch runtime.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Display the configuration of the batch runtime by using the `list-batch-runtime-configuration` subcommand.
3. If desired, use the `get` subcommand to view the attributes of the data source and managed executor service resources.

For example (output omitted):

```
asadmin> get resources.jdbc-resource.jdbc/_TimerPool.*  
...  
asadmin> get resources.managed-executor-  
service.concurrent/_defaultManagedExecutorService.*  
...
```

### Example 10-4 Listing the Batch Runtime Configuration

This example lists the configuration of the batch runtime for the default server instance, `server`.

```
asadmin> list-batch-runtime-configuration
```

```
DATASOURCELOOKUPNAME      EXECUTORSERVICELOOKUPNAME
jdbc/_TimerPool           concurrent/_defaultManagedExecutorService
Command list-batch-runtime-configuration executed successfully.
```

## See Also

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

## To Configure the Batch Runtime

Use the `set-batch-runtime-configuration` subcommand in remote mode to configure the batch runtime.



Do not change the data source after the first batch job has been submitted to the batch runtime for execution. If the data source must be changed, stop and restart the domain and then make the change before any jobs are started or restarted. However, once the data source has been changed, information stored in the previous data source becomes inaccessible.

The managed executor service can be changed after a batch job has been submitted to the batch runtime without affecting execution of the job.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Configure the batch runtime by using the `set-batch-runtime-configuration` subcommand.

### Example 10-5 Configuring the Batch Runtime

This example configures the batch runtime for the default server instance, `server`, to use an existing managed executor service named `concurrent/Executor1`.

```
asadmin> set-batch-runtime-configuration --executorservicelookupname
concurrent/Executor1
Command set-batch-runtime-configuration executed successfully.
```

## See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help set-batch-runtime-configuration` at the command line.

## **Part II**

# **Resources and Services Administration**

# 11 Administering Database Connectivity

This chapter provides procedures for performing database connectivity tasks in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Database Connectivity](#)
- [Setting Up the Database](#)
- [Configuring Access to the Database](#)
- [Configuration Specifics for JDBC Drivers](#)

Instructions for accomplishing these tasks by using the Administration Console are contained in the Administration Console online help.

## About Database Connectivity

A database management system (DBMS) provides facilities for storing, organizing, and retrieving data. The information in databases is often described as persistent data because it is saved on disk and exists after the application process ends. Most business applications store data in relational databases. Applications can access database information by using the Java Database Connectivity (JDBC) API.

The key elements of database connectivity are the following:

- Database. The repository where data is stored for an enterprise. Java EE applications access relational databases through the JDBC API. For administration procedures, see [Setting Up the Database](#).
- JDBC Connection Pool. A JDBC connection pool is a group of reusable connections for a particular database. For administration procedures, see [Administering JDBC Connection Pools](#).
- JDBC Resource. A JDBC resource (data source) provides applications with a means of connecting to a database. To create a JDBC resource, specify the connection pool with which it is associated. Multiple JDBC resources can specify a single connection pool. A JDBC resource is identified by its Java Naming and Directory Interface (JNDI) name. For administration procedures, see [Administering JDBC Resources](#).
- JDBC Driver. A database driver is a software component that enables a Java application to interact with a database connectivity API . Each database requires its own driver. For administration procedures, see [Integrating the JDBC Driver](#).

At runtime, the following sequence occurs when an application connects to a database:

1. The application gets the JDBC resource associated with the database by making a call through the JNDI API.

Using the JNDI name of the resource, the naming and directory service locates the JDBC

resource. Each JDBC resource specifies a connection pool.

2. Using the JDBC resource, the application gets a database connection.

Eclipse GlassFish retrieves a physical connection from the connection pool that corresponds to the database. The pool defines connection attributes such as the database name (URL), user name, and password.

3. After the database connection is established, the application can read, modify, and add data to the database.

The application accesses the database by making calls to the JDBC API. The JDBC driver translates the application's JDBC calls into the protocol of the database server.

4. When the application is finished accessing the database, the application closes the connection and returns the connection to the connection pool.

## Setting Up the Database

Most applications use relational databases to store, organize, and retrieve data. Applications access relational databases through the Java Database Connectivity (JDBC) API.

The following topics are addressed here:

- [To Install the Database and Database Driver](#)
- [To Start the Database](#)
- [To Stop the Database](#)
- [Apache Derby Database Utility Scripts](#)

### To Install the Database and Database Driver

1. Install a supported database product.

To see the current list of database products supported by Eclipse GlassFish, refer to the [Eclipse GlassFish Release Notes](#).

2. Install a supported JDBC driver for the database product.

For a list of drivers supported by Eclipse GlassFish, see [Configuration Specifics for JDBC Drivers](#).

3. Make the JDBC driver JAR file accessible to the domain administration server (DAS).

See [Integrating the JDBC Driver](#).

4. Create the database.

The application provider usually delivers scripts for creating and populating the database.

## Next Steps

You are now ready to create a connection pool for the database, and a JDBC resource that points to the connection pool. See [To Create a JDBC Connection Pool](#) and [To Create a JDBC Resource](#). The final step is to integrate the JDBC driver into an administrative domain as described in [Integrating the JDBC Driver](#).

## To Start the Database

Eclipse GlassFish includes an implementation of the Apache Derby database, however, you can use any JDBC-compliant database. The database is not started automatically when you start Eclipse GlassFish, so if you have applications that require a database, you need to start Apache Derby database manually by using the local `start-database` subcommand.

Start the database by using the `start-database` subcommand.

When the database server starts, or a client connects to it successfully, the following files are created at the location that is specified by the `--dbhome` option:

- The `derby.log` file contains the database server process log along with its standard output and standard error information.
- The database files contain your schema (for example, database tables).

### Example 11-1 Starting a Database

This example starts the Apache Derby database on the host host1 and port 5001. [source]

```
asadmin> start-database --dbhost host1 --dbport 5001 --terse=true
Starting database in the background.
Log redirected to /opt/SUNWappserver/databases/javadb.log.
Command start-database executed successfully.
```

### See Also

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

## To Stop the Database

Use the local `stop-database` subcommand to stop the Apache Derby database on a specified port. A single host can have multiple database server processes running on different ports.

1. If necessary, notify users that the database is being stopped.

2. Stop the database by using the `stop-database` subcommand.

#### Example 11-2 Stopping a Database

This example stops the Apache Derby database on port 5001 of `localhost`.

```
asadmin> stop-database --dbhost=localhost --dbport=5001
onnection obtained for host: localhost, port number 5001.
Apache Derby Network Server - 10.2.2.1 - (538595) shutdown
at 2008-10-17 23:34:2 7.218 GMT
Command stop-database executed successfully.
```

#### Troubleshooting

For a laptop that roams between networks, you might have trouble shutting down the database. If you start the Apache Derby database and then change your IP address, you will not be able to stop the Apache Derby database unless you add a specific `--dbhost` argument. For example, if you run `asadmin start-database dbhost = 0.0.0.0`, and then disconnect Ethernet and switch to wifi, you should run a command similar to the following to stop the database:

```
asadmin stop-database dbhost localhost
```

#### See Also

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

## Apache Derby Database Utility Scripts

The Apache Derby database configuration that is available for use with Eclipse GlassFish includes scripts that can help you use the Apache Derby database. The following scripts are available in the `as-install/javadb/bin` directory:

**startNetworkServer, startNetworkServer.bat**

Script to start the network server

**stopNetworkServer, stopNetworkServer.bat**

Script to stop the network server

**ij, ij.bat**

Interactive JDBC scripting tool

**dblook, dblook.bat**

Script to view all or part of the DDL for a database

**sysinfo, sysinfo.bat**

Script to display versioning information about the Apache Derby database environment

## **NetworkServerControl, NetworkServerControl.bat**

Script to execute commands on the **NetworkServerControl** API

### **To Configure Your Environment to Run the Apache Derby Database Utility Scripts**

1. Ensure that the **JAVA\_HOME** environment variable specifies the directory where the JDK is installed.
2. Set the **JAVADB\_HOME** environment variable to point to the `as-install/javadb` directory.

#### See Also

For more information about these utilities, see the following documentation:

- [Apache Derby Tools and Utilities Guide](#)
- [Derby Server and Administration Guide](#)

## **Configuring Access to the Database**

After establishing the database, you are ready to set up access for Eclipse GlassFish applications. The high-level steps include creating a JDBC connection pool, creating a JDBC resource for the connection pool, and integrating a JDBC driver into an administrative domain.

Instructions for performing these steps are contained in the following sections:

- [Administering JDBC Connection Pools](#)
- [Administering JDBC Resources](#)
- [Enabling the `jdbc/\_default` Resource in a Clustered Environment](#)
- [Integrating the JDBC Driver](#)

## **Administering JDBC Connection Pools**

A JDBC connection pool is a group of reusable connections for a particular database. Because creating each new physical connection is time consuming, Eclipse GlassFish maintains a pool of available connections. When an application requests a connection, it obtains one from the pool. When an application closes a connection, the connection is returned to the pool. JDBC connection pools can be globally accessible or be scoped to an enterprise application, web module, EJB module, connector module or application client module, as described in "[Application-Scope Resources](#)" in Eclipse GlassFish Application Deployment Guide.

A JDBC resource is created by specifying the connection pool with which the resource is associated. Multiple JDBC resources can specify a single connection pool. The properties of connection pools can vary with different database vendors. Some common properties are the database name (URL),

the user name, and the password.

The following tasks and information are used to administer JDBC connection pools:

- [To Create a JDBC Connection Pool](#)
- [To List JDBC Connection Pools](#)
- [To Contact \(Ping\) a Connection Pool](#)
- [To Reset \(Flush\) a Connection Pool](#)
- [To Update a JDBC Connection Pool](#)
- [To Delete a JDBC Connection Pool](#)
- [Configuring Specific JDBC Connection Pool Features](#)

## To Create a JDBC Connection Pool

Use the `create-jdbc-connection-pool` subcommand in remote mode to register a new JDBC connection pool with the specified JDBC connection pool name. A JDBC connection pool or a connector connection pool can be created with authentication. You can either use a subcommand option to specify user, password, or other connection information using the `asadmin` utility, or specify the connection information in the XML descriptor file.

One connection pool is needed for each database, possibly more depending on the application. When you are building the connection pool, certain data specific to the JDBC driver and the database vendor is required. You can find some of the following specifics in [Configuration Specifics for JDBC Drivers](#):

- Database vendor name
- Resource type, such as `javax.sql.DataSource` (local transactions only) `javax.sql.XADatasource` (global transactions)
- Data source class name
- Required properties, such as the database name (URL), user name, and password

Creating a JDBC connection pool is a dynamic event and does not require server restart. However, there are some parameters that do require server restart. See [Configuration Changes That Require Restart](#).

### Before You Begin

Before creating the connection pool, you must first install and integrate the database and its associated JDBC driver. For instructions, see [Setting Up the Database](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create the JDBC connection pool by using the `create-jdbc-connection-pool` subcommand.
3. If needed, restart the server.

Some parameters require server restart. See [Configuration Changes That Require Restart](#).

### Example 11-3 Creating a JDBC Connection Pool

This example creates a JDBC connection pool named `sample_derby_pool` on `localhost`.

```
asadmin> create-jdbc-connection-pool
--datasourceclassname org.apache.derby.jdbc.ClientDataSource
--restype javax.sql.XADatasource
--property portNumber=1527:password=APP:user=APP:serverName=
localhost:databaseName=sun-appserv-samples:connectionAttribut
es=\;create\=true sample_derby_pool
Command create-jdbc-connection-pool executed successfully.
```

#### See Also

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

### To List JDBC Connection Pools

Use the `list-jdbc-connection-pools` subcommand in remote mode to list all existing JDBC connection pools.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the JDBC connection pools by using the `list-jdbc-connection-pools` subcommand.

### Example 11-4 Listing JDBC Connection Pools

This example lists the JDBC connection pools that are on `localhost`.

```
asadmin> list-jdbc-connection-pools
sample_derby_pool2
poolA
__TimerPool
DerbyPool
sample_derby_pool
Command list-jdbc-connection-pools executed successfully.
```

#### See Also

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

## To Contact (Ping) a Connection Pool

Use the `ping-connection-pool` subcommand in remote mode to test if a connection pool is usable. For example, if you create a new JDBC connection pool for an application that is expected to be deployed later, you can test the JDBC pool with this subcommand before the application is deployed. Running a ping will force the creation of the pool if it hasn't already been created.

### Before You Begin

Before you can contact a connection pool, the connection pool must be created with authentication, and the server or database must be running.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Ping a connection pool by using the `ping-connection-pool` subcommand.

### Example 11-5 Contacting a Connection Pool

This example tests to see if the `DerbyPool` connection pool is usable.

```
asadmin> ping-connection-pool DerbyPool
Command ping-connection-pool executed successfully
```

### See Also

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

You can also specify that a JDBC connection pool is automatically tested when created or reconfigured by setting its `--ping` option to `true` (the default is `false`). See [To Create a JDBC Connection Pool](#) or [To Update a JDBC Connection Pool](#).

## To Reset (Flush) a Connection Pool

Use the `flush-connection-pool` in remote mode to reinitialize all connections established in the specified connection pool without the need for reconfiguring the pool. Connection pool reconfiguration can result in application redeployment, which is a time-consuming operation. The JDBC connection pool or connector connection pool is reset to its initial state. Any existing live connections are destroyed, which means that the transactions associated with these connections are lost and must be retried. The subcommand then recreates the initial connections for the pool, and restores the pool to its steady pool size.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Reset a connection pool by using the `flush-connection-pool` subcommand.

### Example 11-6 Resetting (Flushing) a Connection Pool

This example resets the JDBC connection pool named `__TimerPool` to its steady pool size.

```
asadmin> flush-connection-pool __TimerPool  
Command flush-connection-pool executed successfully.
```

## See Also

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

## To Update a JDBC Connection Pool

You can change all of the settings for an existing pool except its name. Use the `get` and `set` subcommands to view and change the values of the JDBC connection pool properties.

1. List the JDBC connection pools by using the `list-jdbc-connection-pools` subcommand.
2. View the attributes of the JDBC connection pool by using the `get` subcommand.

For example:

```
asadmin get resources.jdbc-connection-pool.DerbyPool.property
```

3. Set the attribute of the JDBC connection pool by using the `set` subcommand.

For example:

```
asadmin set resources.jdbc-connection-pool.DerbyPool.steady-pool-size=9
```

4. If needed, restart the server.

Some parameters require server restart. See [Configuration Changes That Require Restart](#).

## See Also

For information about how to tune a connection pool, see the [Eclipse GlassFish Performance Tuning Guide](#).

## To Delete a JDBC Connection Pool

Use the `delete-jdbc-connection-pool` subcommand in remote mode to delete an existing JDBC connection pool. Deleting a JDBC connection pool is a dynamic event and does not require server restart.

### Before You Begin

Before deleting a JDBC connection pool, all associations to the resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the JDBC connection pools by using the `list-jdbc-connection-pools` subcommand.
3. If necessary, notify users that the JDBC connection pool is being deleted.
4. Delete the connection pool by using the `delete-jdbc-connection-pool` subcommand.

#### Example 11-7 Deleting a JDBC Connection Pool

This example deletes the JDBC connection pool named `DerbyPool`.

```
asadmin> delete-jdbc-connection-pool jdbc/DerbyPool  
Command delete-jdbc-connection-pool executed successfully.
```

#### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-jdbc-connection-pool` at the command line.

## Configuring Specific JDBC Connection Pool Features

In Eclipse GlassFish, JDBC Connection Pools support a variety of features to simplify administration, monitoring and performance tuning. The following topics address several of these features:

- [Transparent Pool Reconfiguration](#)
- [Using an Initialization Statement](#)
- [Setting a Statement Timeout](#)
- [Statement Leak Detection and Leaked Statement Reclamation](#)
- [Statement Caching](#)
- [Statement Tracing](#)

### Transparent Pool Reconfiguration

When the properties or attributes of a JDBC connection pool are changed, the connection pool is destroyed and re-created. Normally, applications using the connection pool must be redeployed as a consequence. This restriction can be avoided by enabling transparent JDBC connection pool reconfiguration. When this feature is enabled, applications do not need to be redeployed. Instead, requests for new connections are blocked until the reconfiguration operation completes. Connection requests from any in-flight transactions are served using the old pool configuration so as to complete the transaction. Then, connections are created using the pool's new configuration, and any blocked connection requests are served with connections from the re-created pool.

To enable transparent JDBC connection pool reconfiguration, set the `dynamic-reconfiguration-wait-timeout-in-seconds` property of the JDBC connection pool to a positive, nonzero value in one of the

following ways:

- Add it as a property in the Edit JDBC Connection Pool Properties page in the Administration Console. For more information, click the Help button in the Administration Console.
- Specify it using the `--property` option in the `create-jdbc-connection-pool` subcommand. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Set it using the `set` subcommand. For example:

```
asadmin set resources.jdbc-connection-pool.pool-name.property.dynamic-reconfiguration-wait-timeout-in-seconds=15
```

This property specifies the time in seconds to wait for in-use connections to close and in-flight transactions to complete. Any connections in use or transaction in flight past this time must be retried.

## Using an Initialization Statement

You can specify a statement that executes each time a physical connection to the database is created (not reused) from a JDBC connection pool. This is useful for setting request or session specific properties and is suited for homogeneous requests in a single application. Set the Init SQL attribute of the JDBC connection pool to the SQL string to be executed in one of the following ways:

- Enter an Init SQL value in the Edit Connection Pool Advanced Attributes page in the Administration Console. For more information, click the Help button in the Administration Console.
- Specify the `--initsql` option in the `asadmin create-jdbc-connection-pool` command. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Specify the `init-sql` option in the `asadmin set` command. For example:

```
asadmin set domain1.resources.jdbc-connection-pool.DerbyPool.init-sql="sql-string"
```

## Setting a Statement Timeout

An abnormally long running JDBC query executed by an application may leave it in a hanging state unless a timeout is explicitly set on the statement. Setting a statement timeout guarantees that all queries automatically time out if not completed within the specified period. When statements are created, the `queryTimeout` is set according to the statement timeout setting. This works only when the underlying JDBC driver supports `queryTimeout` for `Statement`, `PreparedStatement`, `CallableStatement`, and `ResultSet`.

You can specify a statement timeout in the following ways:

- Enter a Statement Timeout value in the Edit Connection Pool Advanced Attributes page in the

Administration Console. For more information, click the Help button in the Administration Console.

- Specify the `--statementtimeout` option in the `asadmin create-jdbc-connection-pool` command. For more information, see [create-jdbc-connection-pool\(1\)](#).

## Statement Leak Detection and Leaked Statement Reclamation

If statements are not closed by an application after use, it is possible for the application to run out of cursors. Enabling statement leak detection causes statements to be considered as leaked if they are not closed within a specified period. Additionally, leaked statements can be reclaimed automatically.

To enable statement leak detection, set Statement Leak Timeout In Seconds for the JDBC connection pool to a positive, nonzero value in one of the following ways:

- Specify the `--statementleaktimeout` option in the `create-jdbc-connection-pool` subcommand. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Specify the `statement-leak-timeout-in-seconds` option in the `set` subcommand. For example:

```
asadmin set resources.jdbc-connection-pool.pool-name.statement-leak-timeout-in-seconds=300
```

When selecting a value for Statement Leak Timeout In Seconds, make sure that:

- It is less than the Connection Leak Timeout; otherwise, the connection could be closed before the statement leak is recognized.
- It is greater than the Statement Timeout; otherwise, a long running query could be mistaken as a statement leak.

After enabling statement leak detection, enable leaked statement reclamation by setting Reclaim Leaked Statements for the JDBC connection pool to a `true` value in one of the following ways:

- Specify the `--statementleakreclaim=true` option in the `create-jdbc-connection-pool` subcommand. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Specify the `statement-leak-reclaim` option in the `set` subcommand. For example:

```
asadmin set resources.jdbc-connection-pool.pool-name.statement-leak-reclaim=true
```

## Statement Caching

Statement caching stores statements, prepared statements, and callable statements that are executed repeatedly by applications in a cache, thereby improving performance. Instead of the statement being prepared each time, the cache is searched for a match. The overhead of parsing

and creating new statements each time is eliminated.

Statement caching is usually a feature of the JDBC driver. The Eclipse GlassFish provides caching for drivers that do not support caching. To enable this feature, set the Statement Cache Size for the JDBC connection pool in one of the following ways:

- Enter a Statement Cache Size value in the Edit Connection Pool Advanced Attributes page in the Administration Console. For more information, click the Help button in the Administration Console.
- Specify the `--statementcachesize` option in the `asadmin create-jdbc-connection-pool` command. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Specify the `statement-cache-size` option in the `asadmin set` command. For example:

```
asadmin set domain1.resources.jdbc-connection-pool.DerbyPool.statement-cache-size=10
```

By default, this attribute is set to zero and the statement caching is turned off. To enable statement caching, you can set any positive nonzero value. The built-in cache eviction strategy is LRU-based (Least Recently Used). When a connection pool is flushed, the connections in the statement cache are recreated.

## Statement Tracing

You can trace the SQL statements executed by applications that use a JDBC connection pool. Set the SQL Trace Listeners attribute to a comma-separated list of trace listener implementation classes in one of the following ways:

- Enter an SQL Trace Listeners value in the Edit Connection Pool Advanced Attributes page in the Administration Console. For more information, click the Help button in the Administration Console.
- Specify the `--sqltracelisteners` option in the `asadmin create-jdbc-connection-pool` command. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Specify the `sql-trace-listeners` option in the `asadmin set` command. For example:

```
asadmin set domain1.resources.jdbc-connection-pool.DerbyPool.sql-trace-listeners=listeners
```

The Eclipse GlassFish provides a public interface, `org.glassfish.api.jdbc.SQLTraceListener`, that implements a means of recording `SQLTraceRecord` objects. To make custom implementations of this interface available to the Eclipse GlassFish, place the implementation classes in `as-install/lib`.

The Eclipse GlassFish provides an SQL tracing logger to log the SQL operations in the form of `SQLTraceRecord` objects in the `server.log` file. The module name under which the SQL operation is logged is `jakarta.enterprise.resource.sqltrace`. SQL traces are logged as FINE messages along with

the module name to enable easy filtering of the SQL logs. A sample SQL trace record looks like this:

```
[#|2009-11-  
27T15:46:52.202+0530|FINE|glassfish7.0|jakarta.enterprise.resource.sqltrace.com.sun.gj  
c.util  
|_ThreadID=29;_ThreadName=Thread-  
1;ClassName=com.sun.gjc.util.SQLTraceLogger;MethodName=sqlTrace;  
|ThreadID=77 | ThreadName=p: thread-pool-1; w: 6 | TimeStamp=1259317012202  
| ClassName=com.sun.gjc.spi.jdbc40.PreparedStatementWrapper40 |  
MethodName=executeUpdate  
| arg[0]=insert into table1(colName) values(100) | arg[1]=columnNames | |#]
```

This trace shows that an `executeUpdate(String sql, String columnNames)` operation is being done.

When SQL statement tracing is enabled and JDBC connection pool monitoring is enabled, Eclipse GlassFish maintains a tracing cache of recent queries and their frequency of use. The following JDBC connection pool properties can be configured to control this cache and the monitoring statistics available from it:

#### **time-to-keep-queries-in-minutes**

Specifies how long in minutes to keep a query in the tracing cache, tracking its frequency of use. The default value is 5 minutes.

#### **number-of-top-queries-to-report**

Specifies how many of the most used queries, in frequency order, are listed the monitoring report. The default value is 10 queries.

Set these parameters in one of the following ways:

- Add them as properties in the Edit JDBC Connection Pool Properties page in the Administration Console. For more information, click the Help button in the Administration Console.
- Specify them using the `--property` option in the `create-jdbc-connection-pool` subcommand. For more information, see [create-jdbc-connection-pool\(1\)](#).
- Set them using the `set` subcommand. For example:

```
asadmin set resources.jdbc-connection-pool.pool-name.property.time-to-keep-queries-  
in-minutes=10
```

## **Administering JDBC Resources**

A JDBC resource, also known as a data source, provides an application with a means of connecting to a database. Typically, you create a JDBC resource for each database that is accessed by the applications deployed in a domain. Multiple JDBC resources can be specified for a database. JDBC resources can be globally accessible or be scoped to an enterprise application, web module, EJB module, connector module or application client module, as described in "[Application-Scoped](#)

[Resources](#)" in Eclipse GlassFish Application Deployment Guide.

A JDBC resource is created by specifying the connection pool with which the resource will be associated . Use a unique Java Naming and Directory Interface (JNDI) name to identify the resource. For example, the JNDI name for the resource of a payroll database might be `java:comp/env/jdbc/payrolldb`.

The Jakarta EE standard specifies that certain default resources be made available to applications, and defines specific JNDI names for these default resources. Eclipse GlassFish makes these names available through the use of logical JNDI names, which map Jakarta EE standard JNDI names to specific Eclipse GlassFish resources. For JDBC resources, the Jakarta EE standard name `java:comp/DefaultDataSource` is mapped to the `jdbc/_default` resource.

The following tasks and information are used to administer JDBC resources:

- [To Create a JDBC Resource](#)
- [To List JDBC Resources](#)
- [To Update a JDBC Resource](#)
- [To Delete a JDBC Resource](#)

## To Create a JDBC Resource

Use the `create-jdbc-resource` subcommand in remote mode to create a JDBC resource. Creating a JDBC resource is a dynamic event and does not require server restart.

Because all JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a JDBC resource in the Administration Console, use only the `jdbc/'name` format. For example, a payroll database might be specified as `'jdbc/payrolldb`.

### Before You Begin

Before creating a JDBC resource, you must first create a JDBC connection pool. For instructions, see [To Create a JDBC Connection Pool](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a JDBC resource by using the `create-jdbc-resource` subcommand.

Information about properties for the subcommand is included in this help page.

3. If necessary, notify users that the new resource has been created.

### Example 11-8 Creating a JDBC Resource

This example creates a JDBC resource named `DerbyPool`.

```
asadmin> create-jdbc-resource --connectionpoolid DerbyPool jdbc/DerbyPool
Command create-jdbc-resource executed successfully.
```

## See Also

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

## To List JDBC Resources

Use the `list-jdbc-resources` subcommand in remote mode to list the existing JDBC resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List JDBC resources by using the `list-jdbc-resources` subcommand.

### Example 11-9 Listing JDBC Resources

This example lists JDBC resources for `localhost`.

```
asadmin> list-jdbc-resources
jdbc/_TimerPool
jdbc/DerbyPool
jdbc/_default
jdbc1
Command list-jdbc-resources executed successfully.
```

## See Also

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

## To Update a JDBC Resource

You can enable or disable a JDBC resource by using the `set` subcommand. The JDBC resource is identified by its dotted name.

1. List JDBC resources by using the `list-jdbc-resources` subcommand.
2. Modify the values for the specified JDBC resource by using the `set` subcommand.

For example:

### Example 11-10 Updating a JDBC Resource

This example changes the `res1` enabled setting to false.

```
asadmin>set resources.jdbc-resource.res1.enabled=false
```

## To Delete a JDBC Resource

Use the `delete-jdbc-resource` subcommand in remote mode to delete an existing JDBC resource. Deleting a JDBC resource is a dynamic event and does not require server restart.

### Before You Begin

Before deleting a JDBC resource, all associations with this resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List JDBC resources by using the `list-jdbc-resources` subcommand.
3. If necessary, notify users that the JDBC resource is being deleted.
4. Delete a JDBC resource by using the `delete-jdbc-resource` subcommand.

### Example 11-11 Deleting a JDBC Resource

This example deletes a JDBC resource named `DerbyPool`.

```
asadmin> delete-jdbc-resource jdbc/DerbyPool  
Command delete-jdbc-resource executed successfully.
```

### See Also

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

## Enabling the `jdbc/_default` Resource in a Clustered Environment

Eclipse GlassFish 7 includes a preconfigured JDBC resource with the JNDI name `jdbc/default`. This `jdbc/default` resource is not enabled by default, so you need to explicitly enable it if you want to use it in a cluster.

### To Enable the `jdbc/_default` Resource for a Clustered Environment

Instructions for creating JDBC resources in general are provided in [To Create a JDBC Resource](#). Use the following procedure to enable the preconfigured `jdbc/_default` resource for a clustered Eclipse GlassFish environment.

1. Create the `jdbc/_default` resource reference for the cluster.

```
asadmin create-resource-ref --target cluster-name jdbc/_default
```

2. Enable the resource on the DAS that manages the cluster.

```
asadmin set resources.jdbc-connection-pool.DerbyPool.property.serverName=DAS-machine-name
```

This step is only required if the cluster includes remote instances.

3. Restart the DAS and the target cluster(s).

```
asadmin stop-cluster cluster-name  
asadmin stop-domain domain-name  
asadmin start-domain domain-name  
asadmin start-cluster cluster-name
```

## Integrating the JDBC Driver

To use JDBC features, you must choose a JDBC driver to work with the Eclipse GlassFish, then you must set up the driver. This section covers these topics:

- [Supported Database Drivers](#)
- [Making the JDBC Driver JAR Files Accessible](#)
- [Automatic Detection of Installed Drivers](#)

### Supported Database Drivers

Supported JDBC drivers are those that have been fully tested by Oracle. For a list of the JDBC drivers currently supported by the Eclipse GlassFish, see the [Eclipse GlassFish Release Notes](#). For configurations of supported and other drivers, see [Configuration Specifics for JDBC Drivers](#).



Because the drivers and databases supported by the Eclipse GlassFish are constantly being updated, and because database vendors continue to upgrade their products, always check with Oracle technical support for the latest database support information.

### Making the JDBC Driver JAR Files Accessible

To integrate the JDBC driver into a Eclipse GlassFish domain, copy the JAR files into the domain-dir/`lib` directory, then restart the server. This makes classes accessible to all applications or modules deployed on servers that share the same configuration. For more information about Eclipse GlassFish class loaders, see "[Class Loaders](#)" in Eclipse GlassFish Application Development Guide.

If you are using an Oracle database with EclipseLink extensions, copy the JAR files into the domain-

dir/lib/ext directory, then restart the server. For details, see "Oracle Database Enhancements" in Eclipse GlassFish Application Development Guide.

## Automatic Detection of Installed Drivers

The Administration Console detects installed JDBC Drivers automatically when you create a JDBC connection pool. To create a JDBC connection pool using the Administration Console, open the Resources component, open the JDBC component, select Connection Pools, and click on the New button. This displays the New JDBC Connection Pool page.

Based on the Resource Type and Database Vendor you select on the New JDBC Connection Pool page, data source or driver implementation class names are listed in the Datasource Classname or Driver Classname field when you click on the Next button. When you choose a specific implementation class name on the next page, additional properties relevant to the installed JDBC driver are displayed in the Additional Properties section.

## Configuration Specifics for JDBC Drivers

Eclipse GlassFish is designed to support connectivity to any database management system by using a corresponding JDBC driver. Configuration information is provided for these JDBC drivers:

- [IBM DB2 Database Type 2 Driver](#)
- [IBM DB2 Database Type 4 Driver](#)
- [Apache Derby DB/Derby Type 4 Driver](#)
- [MySQL Server Database Type 4 Driver](#)
- [Oracle 10 Database Driver](#)
- [Oracle 11 Database Driver](#)
- [PostgreSQL Type 4 Driver](#)
- [DataDirect Type 4 Driver for IBM DB2 Database](#)
- [DataDirect Type 4 Driver for IBM Informix](#)
- [DataDirect Type 4 Driver for Microsoft SQL Server Database](#)
- [DataDirect Type 4 Driver for MySQL Server Database](#)
- [DataDirect Type 4 Driver for Oracle 11 Database](#)
- [DataDirect Type 4 Driver for Sybase Database](#)
- [Inet Oraxo Driver for Oracle Database](#)
- [Inet Merlia Driver for Microsoft SQL Server Database](#)
- [Inet Sybelux Driver for Sybase Database](#)
- [JConnect Type 4 Driver for Sybase ASE 12.5 Database](#)

## IBM DB2 Database Type 2 Driver

The JAR files for the DB2 driver are `db2jcc.jar`, `db2jcc_license_cu.jar`, and `db2java.zip`. Set your environment variables . For example:

```
LD_LIBRARY_PATH=/usr/db2user/sqllib/lib:${Jakarta EE.home}/lib  
DB2DIR=/opt/IBM/db2/V8.2  
DB2INSTANCE=db2user  
INSTHOME=/usr/db2user  
VWSPATH=/usr/db2user/sqllib  
THREADS_FLAG=native
```

Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DB2
- DataSource Classname: `com.ibm.db2.jcc.DB2SimpleDataSource`
- Properties:
  - `databaseName` - Set as appropriate.
  - `user` - Set as appropriate.
  - `password` - Set as appropriate.
  - `driverType` - Set to `2`.
  - `deferPrepares` - Set to `false`.

## IBM DB2 Database Type 4 Driver

The JAR file for the DB2 driver is `db2jcc.jar`. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DB2
- DataSource Classname: `com.ibm.db2.jcc.DB2SimpleDataSource`
- Properties:
  - `databaseName` - Set as appropriate.
  - `user` - Set as appropriate.
  - `password` - Set as appropriate.

- **driverType** - Set to 4.

## Apache Derby DB/Derby Type 4 Driver

The Apache Derby DB/Derby JDBC driver is included with Eclipse GlassFish by default, so you do not need to integrate this JDBC driver with Eclipse GlassFish.

The JAR file for the Apache Derby DB driver is [derbyclient.jar](#). Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: Apache Derby
- DataSource Classname: Specify one of the following:

```
org.apache.derby.jdbc.ClientDataSource40  
org.apache.derby.jdbc.ClientXADataSource40
```

- Properties:

- **serverName** - Specify the host name or IP address of the database server.
- **portNumber** - Specify the port number of the database server if it is different from the default.
- **databaseName** - Specify the name of the database.
- **user** - Specify the database user.

This is only necessary if the Apache Derby database is configured to use authentication. The Apache Derby database does not use authentication by default. When the user is provided, it is the name of the schema where the tables reside.

- **password** - Specify the database password.

This is only necessary if the Apache Derby database is configured to use authentication.

## MySQL Server Database Type 4 Driver

The JAR file for the MySQL driver is [mysql-connector-java-5.1.14-bin.jar](#). Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: MySql
- DataSource Classname:

```
com.mysql.jdbc.jdbc2.optional.MysqlDataSource  
com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
```

- Properties:

- **serverName** - Specify the host name or IP address of the database server.
- **portNumber** - Specify the port number of the database server.
- **databaseName** - Set as appropriate.
- **user** - Set as appropriate.
- **password** - Set as appropriate.

## Oracle 10 Database Driver

The JAR file for the Oracle 10 database driver is **ojdbc14.jar**. Make sure that the shared library is available through **LD\_LIBRARY\_PATH** and that the **ORACLE\_HOME** property is set.

To make the Oracle driver behave in a Jakarta EE-compliant manner, you must define the following JVM property:

```
-Doracle.jdbc.J2EE13Compliant=true
```

Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: Oracle
- DataSource Classname: Specify one of the following:

```
oracle.jdbc.pool.OracleDataSource  
oracle.jdbc.xa.client.OracleXADatasource
```

- Properties:

- **user** - Set as appropriate.
- **password** - Set as appropriate.

## Oracle 11 Database Driver

The JAR file for the Oracle 11 database driver is **ojdbc6.jar**.

To make the Oracle driver behave in a Jakarta EE-compliant manner, you must define the following

JVM property:

```
-Doracle.jdbc.J2EE13Compliant=true
```

Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: Oracle
- DataSource Classname: Specify one of the following:

```
oracle.jdbc.pool.OracleDataSource  
oracle.jdbc.xa.client.OracleXADataSource
```

- Properties:
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.

For this driver, the `XAResource.recover` method repeatedly returns the same set of in-doubt Xids regardless of the input flag. According to the XA specifications, the Transaction Manager initially calls this method with `TMSTARTSCAN` and then with `TMNOFLAGS` repeatedly until no Xids are returned. The `XAResource.commit` method also has some issues.

To disable this Eclipse GlassFish workaround, the `oracle-xa-recovery-workaround` property value must be set to `false`.



Additionally, in order for the transaction manager to recover transactions, the JDBC connection pool's database user must be given certain Oracle permissions:

- SELECT permission on DBA\_PENDING\_TRANSACTIONS, PENDING\_TRANS\$, DBA\_2PC\_PENDING and DBA\_2PC\_NEIGHBORS.
- EXECUTE permissions on DBMS\_XA and DBMS\_SYSTEM.

## PostgreSQL Type 4 Driver

The JAR file for the PostgreSQL driver is `postgresql-9.0-801.jdbc4.jar`. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.

- Database Vendor: Postgresql
- DataSource Classname: `org.postgresql.ds.PGSimpleDataSource`
- Properties:
  - `serverName` - Specify the host name or IP address of the database server.
  - `portNumber` - Specify the port number of the database server.
  - `databaseName` - Set as appropriate.
  - `user` - Set as appropriate.
  - `password` - Set as appropriate.

## DataDirect Type 4 Driver for IBM DB2 Database

The JAR file for DataDirect driver is `db2.jar`. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-DB2
- DataSource Classname: `com.ddtek.jdbcx.db2.DB2DataSource`
- Properties:
  - `serverName` - Specify the host name or IP address of the database server.
  - `portNumber` - Specify the port number of the database server.
  - `databaseName` - Set as appropriate.
  - `user` - Set as appropriate.
  - `password` - Set as appropriate.

## DataDirect Type 4 Driver for IBM Informix

Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-Informix
- DataSource Classname: Specify one of the following:

```
com.informix.jdbcx.IfxDataSource
com.informix.jdbcx.IfxXADataSource
```

DataDirect DataSource Classname: `com.ddtek.jdbcx.informix.InformixDataSource`

- Properties:

- `serverName` - Specify the Informix database server name.
- `portNumber` - Specify the port number of the database server.
- `databaseName` - Set as appropriate. This is optional.
- `user` - Set as appropriate.
- `password` - Set as appropriate.
- `IfxIFXHost` - Specify the host name or IP address of the database server.

## DataDirect Type 4 Driver for Microsoft SQL Server Database

The JAR file for the DataDirect driver is `sqlserver.jar`. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-Microsoft SQL Server
- DataSource Classname: `com.ddtek.jdbcx.sqlserver.SQLServerDataSource`
- Properties:
  - `serverName` - Specify the host name or IP address and the port of the database server.
  - `portNumber` - Specify the port number of the database server.
  - `user` - Set as appropriate.
  - `password` - Set as appropriate.
  - `selectMethod` - Set to `cursor`.

## DataDirect Type 4 Driver for MySQL Server Database

The JAR file for the DataDirect driver is `mysql.jar`. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-MySQL
- DataSource: `com.ddtek.jdbcx.mysql.MySQLDataSource`
- Properties:
  - `serverName` - Specify the host name or IP address and the port of the database server.

- **portNumber** - Specify the port number of the database server.
- **user** - Set as appropriate.
- **password** - Set as appropriate.
- **selectMethod** - Set to **cursor**.

## DataDirect Type 4 Driver for Oracle 11 Database

The JAR file for the DataDirect driver is **oracle.jar**.

To make the Oracle driver behave in a Jakarta EE-compliant manner, you must define the following JVM property:

```
-Doracle.jdbc.J2EE13Compliant=true
```

Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-Oracle
- DataSource Classname: **com.ddtek.jdbcx.oracle.OracleDataSource**
- Properties:
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.

## DataDirect Type 4 Driver for Sybase Database

The JAR file for the DataDirect driver is **sybase.jar**. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: DataDirect-Sybase
- DataSource Classname: **com.ddtek.jdbcx.sybase.SybaseDataSource**
- Properties:
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.

- **databaseName** - Set as appropriate. This is optional.
- **user** - Set as appropriate.
- **password** - Set as appropriate.

 In some situations, using this driver can cause exceptions to be thrown because the driver creates a stored procedure for every parameterized PreparedStatement by default. If this situation arises, add the property **PrepareMethod**, setting its value to **direct**.

## Inet Oraxo Driver for Oracle Database

The JAR file for the Inet Oracle driver is **Oranxo.jar**. Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: Oracle
- DataSource Classname: **com.inet.ora.OraDataSource**
- Properties:
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **user** - Specify the database user.
  - **password** - Specify the database password.
  - **serviceName** - Specify the URL of the database. The syntax is as follows:

```
jdbc:inetora:server:port:dbname
```

For example:

```
jdbc:inetora:localhost:1521:payrolldb
```

In this example, **localhost** is the name of the host running the Oracle server, **1521** is the Oracle server's port number, and **payrolldb** is the SID of the database. For more information about the syntax of the database URL, see the Oracle documentation.

- **streamstolob** - If the size of BLOB or CLOB data types exceeds 4 KB and this driver is used for CMP, this property must be set to **true**.

## Inet Merlia Driver for Microsoft SQL Server Database

The JAR file for the Inet Microsoft SQL Server driver is [Merlia.jar](#). Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: MicrosoftSqlServer
- DataSource Classname: [com.inet.tds.TdsDataSource](#)
- Properties:
  - **serverName** - Specify the host name or IP address and the port of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.

## Inet Sybelux Driver for Sybase Database

The JAR file for the Inet Sybase driver is [Sybelux.jar](#). Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.
- Database Vendor: Sybase
- DataSource Classname: [com.inet.syb.SybDataSource](#)
- Properties:
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **databaseName** - Set as appropriate. Do not specify the complete URL, only the database name.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.

## JConnect Type 4 Driver for Sybase ASE 12.5 Database

The JAR file for the Sybase driver is [jconn4.jar](#). Configure the connection pool using the following settings:

- Name: Use this name when you configure the JDBC resource later.
- Resource Type: Specify the appropriate value.

- Database Vendor: Sybase
- DataSource Classname: Specify one of the following:

```
com.sybase.jdbc4.jdbc.SybDataSource  
com.sybase.jdbc4.jdbc.SybXADataSource
```

- Properties:
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **databaseName** - Set as appropriate. Do not specify the complete URL, only the database name.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.
  - **BE\_AS\_JDBC\_COMPLIANT\_AS\_POSSIBLE** - Set to **true**.
  - **FAKE\_METADATA** - Set to **true**.

# 12 Administering EIS Connectivity

This chapter provides information and procedures for administering connections to enterprise information system (EIS) data in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.



If you installed the Web Profile, connector modules that use only outbound communication features and work-management that does not involve inbound communication features are supported. Other connector features are supported only in the Full Platform Profile.

The following topics are addressed here:

- [About EIS Connectivity](#)
- [Administering Connector Connection Pools](#)
- [Administering Connector Resources](#)
- [Administering the Resource Adapter Configuration](#)
- [Administering Connector Security Maps](#)
- [Administering Connector Work Security Maps](#)
- [Administering Administered Objects](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

For information about database connectivity, see [Administering Database Connectivity](#).

## About EIS Connectivity

Enterprise information system (EIS) refers to any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application. Connection resources are used by applications and modules to access EIS software.)

The key elements of EIS connectivity are the following:

- **Connector Module.** A connector module, also called a resource adapter, is a Jakarta EE component that enables applications to interact with EIS software. A connector module is used by Eclipse GlassFish to implement Java Message Service (JMS). Like other Jakarta EE modules, a connector module is installed when it is deployed. For instructions on creating a connector module, see "[Developing Connectors](#)" in Eclipse GlassFish Application Development Guide.
- **Connector Connection Pool.** A connector connection pool is a group of reusable connections for a particular EIS. A connector connection pool is created when you specify the connector module that is associated with the pool. For administration procedures, see [Administering Connector Connection Pools](#).

- **Connector Resource.** A connector resource is a program object that provides an application with a connection to an EIS. A connector resource is created when you specify its JNDI name and its associated connection pool. The JNDI name of a connector resource for an EIS is usually in the `java:comp/env/` eis-specific subcontext. For administration procedures, see [Administering Connector Resources](#).
- **Connector Module Configuration.** A connector module configuration is the information that resides in the domain configuration file ([domain.xml](#)) for the particular connector module (resource adapter). For administration procedures, see [Administering the Resource Adapter Configuration](#).
- **Connector Security Map.** A connector security map associates the caller identity of the application (principal or user group) to a suitable EIS principal or group. For administration procedures, see [Administering Connector Security Maps](#).
- **Connector Work Security Map.** A connector work security map associates the caller identity of the work submitted by the connector module (resource adapter) EIS principal or EIS user group to a suitable principal or user group in the Eclipse GlassFish security domain. For administration procedures, see [Administering Connector Work Security Maps](#).
- **Administered Object.** An administered object provides specialized functionality for an application, such as providing access to a parser that is specific to the connector module and its associated EIS. For administration procedures, see [Administering Administered Objects](#).

At runtime, the following sequence occurs when an application connects to an EIS:

1. The application gets the connector resource (data source) associated with the EIS by making a call through the JNDI API.

Using the JNDI name of the connector resource, the naming and directory service locates the resource. Each EIS resource specifies a connector connection pool.

2. Using the connector resource, the application gets an EIS connection.

Eclipse GlassFish retrieves a physical connection from the connection pool that corresponds to the EIS resource. The pool defines connection attributes such as the EIS name, user name, and password.

3. After the EIS connection is established, the application can read, modify, and add data to the EIS.

The application accesses the EIS information by making calls to the JMS API.

4. When the application is finished accessing the EIS, the application closes the connection and returns the connection to the connection pool.

## Administering Connector Connection Pools

After a connector module has been deployed, you are ready to create a connector connection pool for it.

The following topics are addressed here:

- [To Create a Connector Connection Pool](#)
- [To List Connector Connection Pools](#)
- [To Connect to \(Ping\) or Reset \(Flush\) a Connector Connection Pool](#)
- [To Update a Connector Connection Pool](#)
- [To Delete a Connector Connection Pool](#)

## To Create a Connector Connection Pool

Use the `create-connector-connection-pool` subcommand in remote mode to create a connector connection pool for a deployed connector module. When you are building the connector connection pool, certain data specific to the EIS will be required. The value in the mandatory `--connectiondefinition` option provides the EIS info.

Multiple connector resources can specify a single connection pool.

Creating a connector connection pool is a dynamic event and does not require server restart. However, there are some parameters that do require server restart. See [Configuration Changes That Require Restart](#).

### Before You Begin

Before creating the connector connection pool, the connector must be installed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create the connector connection pool by using the `create-connector-connection-pool` subcommand.

Information about properties for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

4. You can verify that a connection pool is usable by using the `ping-connection-pool` subcommand.

For instructions, see [To Contact \(Ping\) a Connection Pool](#).

### Example 12-1 Creating a Connector Connection Pool

This example creates the new `jms/qConnPool` pool for the `jakarta.jms.QueueConnectionFactory` connector module.

```
asadmin> create-connector-connection-pool --steadyPoolSize 20 --maxPoolSize 100  
--poolResize 2 --maxWait 60000 --raname jmsra --connectionDefinition
```

```
jakarta.jms.QueueConnectionFactory jms/qConnPool
```

```
Command create-connector-connection-pool executed successfully
```

## See Also

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

## To List Connector Connection Pools

Use the `list-connector-connection-pools` subcommand in remote mode to list the pools that have been created.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector connection pools by using the `list-connector-connection-pools` subcommand.

### Example 12-2 Listing Connector Connection Pools

This example lists the existing connector connection pools.

```
asadmin> list-connector-connection-pools
jms/qConnPool
Command list-connector-connection-pools executed successfully
```

## See Also

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

## To Connect to (Ping) or Reset (Flush) a Connector Connection Pool

Use the `ping-connection-pool` or `flush-connection-pool` subcommands in remote mode to perform these tasks on a connection pools. See [To Contact \(Ping\) a Connection Pool](#) or [To Reset \(Flush\) a Connection Pool](#) for instructions.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Connect to or reset a connector connection pool by using the `flush-connection-pool` subcommand or the `ping-connection-pool` subcommand.

## To Update a Connector Connection Pool

Use the `get` and `set` subcommands to view and change the values of the connector connection pool properties.

1. List the connector connection pools by using the `list-connector-connection-pools` subcommand.
2. View the properties of the connector connection pool by using the `get` subcommand. For example:

```
asadmin> get domain.resources.connector-connection-pool.conectionpoolname.*
```

3. Set the property of the connector connection pool by using the `set` subcommand. For example:

```
asadmin> set domain.resources.connector-connection-pool  
.conectionpoolname.validate-atmost-once-period-in-seconds=3
```

4. If needed, restart the server. Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

## To Delete a Connector Connection Pool

Use the `delete-connector-connection-pool` subcommand in remote mode to remove a connector connection pool.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector connection pools by using the `list-connector-connection-pools` subcommand.
3. If necessary, notify users that the connector connection pool is being deleted.
4. Delete the connector connection pool by using the `delete-connector-connection-pool` subcommand.

### Example 12-3 Deleting a Connector Connection Pool

This example deletes the connection pool named `jms/qConnPool`.

```
asadmin> delete-connector-connection-pool --cascade=false jms/qConnPool  
Command delete-connector-connection-pool executed successfully
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-connector-connection-pool` at the command line.

# Administering Connector Resources

A connector resource provides an application or module with the means of connecting to an EIS. Typically, you create a connector resource for each EIS that is accessed by the applications deployed in the domain.

The following topics are addressed here:

- [To Create a Connector Resource](#)
- [To List Connector Resources](#)
- [To Update a Connector Resource](#)
- [To Delete a Connector Resource](#)

## To Create a Connector Resource

Use the `create-connector-resource` subcommand in remote mode to register a new connector resource with its JNDI name.

Creating a connector resource is a dynamic event and does not require server restart. However, there are some parameters that do require server restart. See [Configuration Changes That Require Restart](#).

### Before You Begin

Before creating a connector resource, you must first create a connector connection pool. For instructions, see [To Create a Connector Connection Pool](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create the connector resource by using the `create-connector-resource` subcommand.

Information about properties for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 12-4 Creating a Connector Resource

This example creates a new resource named `jms/qConnFactory` for the `jms/qConnPool` connection pool.

```
asadmin> create-connector-resource --poolname jms/qConnPool  
--description "creating sample connector resource" jms/qConnFactory  
Command create-connector-resource executed successfully
```

## See Also

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

## To List Connector Resources

Use the `list-connector-resources` subcommand in remote mode to list the connector resources that have been created.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector connection pools by using the `list-connector-resources` subcommand.

### Example 12-5 Listing Connector Resources

This example lists the existing connector resources.

```
asadmin> list-connector-resources
jms/qConnFactory
Command list-connector-resources executed successfully
```

## See Also

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

## To Update a Connector Resource

Use the `get` and `set` subcommands to view and change the values of the connector resource properties.

1. List the connector connection pools by using the `list-connector-resources` subcommand.
2. View the properties of the connector resource by using the `get` subcommand. For example

```
asadmin> get domain.resources.connector-resource.jms/qConnFactory
```

3. Set the property of the connector resource by using the `set` subcommand. For example:

```
asadmin> set domain.resources.connector-resource.jms/qConnFactory.enabled=true
```

4. If needed, restart the server. Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

## To Delete a Connector Resource

Use the `delete-connector-resource` subcommand in remote mode to remove a connector resource by specifying the JNDI name.

### Before You Begin

Before deleting a resource, all associations with the resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector connection pools by using the `list-connector-resources` subcommand.
3. If necessary, notify users that the connector resource is being deleted.
4. Delete the connector resource by using the `delete-connector-resource` subcommand.

### Example 12-6 Deleting a Connector Resource

This example deletes the `jms/qConnFactory` connector resource.

```
asadmin> delete-connector-resource jms/qConnFactory
Command delete-connector-resources executed successfully
```

### See Also

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

## Administering the Resource Adapter Configuration

The following topics are addressed here:

- [To Create Configuration Information for a Resource Adapter](#)
- [To List Resource Adapter Configurations](#)
- [To Update a Resource Adapter Configuration](#)
- [To Delete a Resource Adapter Configuration](#)

## To Create Configuration Information for a Resource Adapter

Use the `create-resource-adapter-config` subcommand in remote mode to create configuration information for a resource adapter, also known as a connector module. You can run the subcommand before deploying a resource adapter, so that the configuration information is available at the time of deployment. The resource adapter configuration can also be created after

the resource adapter is deployed. In this situation, the resource adapter is restarted with the new configuration.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create configuration information by using the `create-resource-adapter-config` subcommand.

Information about properties for the subcommand is included in this help page.

#### Example 12-7 Creating a Resource Adapter Configuration

This example creates the configuration for resource adapter `ra1`.

```
asadmin> create-resource-adapter-config --property foo=bar  
--threadpoolid mycustomerthreadpool ra1  
Command create-resource-adapter-config executed successfully
```

#### See Also

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

## To List Resource Adapter Configurations

Use the `list-resource-adapter-configs` subcommand in remote mode to list the configuration information contained in the domain configuration file (`domain.xml`) for the specified resource adapter (connector module).

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the configurations for a resource adapter by using the `list-resource-adapter-configs` subcommand.

#### Example 12-8 Listing Configurations for a Resource Adapter

This example lists all the resource adapter configurations.

```
asadmin> list-resource-adapter-configs  
ra1  
ra2  
Command list-resource-adapter-configs executed successfully
```

#### See Also

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

## To Update a Resource Adapter Configuration

Use the `get` and `set` subcommands to view and change the values of the resource adapter configuration properties.

1. List the configurations for a resource adapter by using the `list-resource-adapter-configs` subcommand.
2. View the properties of the connector resource by using the `get` subcommand. For example:

```
asadmin>get domain.resources.resource-adapter-config.ra1.*
```

3. Set the property of the connector resource by using the `set` subcommand. For example:

```
asadmin> set domain.resources.resource-adapter-config.ra1.raSpecificProperty=value
```

## To Delete a Resource Adapter Configuration

Use the `delete-resource-adapter-config` subcommand in remote mode to delete the configuration information contained in the domain configuration file (`domain.xml`) for a specified resource adapter (connector module).

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the configurations for a resource adapter by using the `list-resource-adapter-configs` subcommand.
3. Delete the configuration for a resource adapter by using the `delete-resource-adapter-config` subcommand.

### Example 12-9 Deleting a Resource Adapter Configuration

This example deletes the configuration for resource adapter `ra1`.

```
asadmin> delete-resource-adapter-config ra1  
Command delete-resource-adapter-config executed successfully
```

### See Also

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

# Administering Connector Security Maps

The EIS is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application. The connector security map is used to map the application's credentials to the EIS credentials.

A security map applies to a particular connector connection pool. One or more named security maps can be associated with a connector connection pool.

The following topics are addressed here:

- [To Create a Connector Security Map](#)
- [To List Connector Security Maps](#)
- [To Update a Connector Security Map](#)
- [To Delete a Connector Security Map](#)

## To Create a Connector Security Map

Use the `create-connector-security-map` subcommand in remote mode to create a security map for the specified connector connection pool. If the security map is not present, a new one is created. You can specify back-end EIS principals or back-end EIS user groups. The connector security map configuration supports the use of the wild card asterisk (\*) to indicate all users or all user groups.

You can also use this subcommand to map the caller identity of the application (principal or user group) to a suitable EIS principal in container-managed authentication scenarios.

### Before You Begin

For this subcommand to succeed, you must have first created a connector connection pool. For instructions, see [To Create a Connector Connection Pool](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a connector security map by using the `create-connector-security-map` subcommand.

Information about the options for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 12-10 Creating a Connector Security Map

This example creates a connector security map `securityMap1` for `connection-pool1`.

```
asadmin> create-connector-security-map --poolname connector-pool1  
--principals principal1, principal2 --mappedusername backend-username securityMap1
```

```
Command create-connector-security-map executed successfully
```

## To List Connector Security Maps

Use the `list-connector-security-maps` subcommand in remote mode to list the existing security maps belonging to the specified connector connection pool. You can get a simple listing of the connector security maps for a connector connection pool, or you can get a more comprehensive listing that shows the principals of the map.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List existing connector connection pools by using the `list-connector-connection-pools` subcommand.
3. List the security maps for a specific connector connection pool by using the `list-connector-security-maps` subcommand.

### Example 12-11 Listing All Connector Security Maps for a Connector Connection Pool

This example lists the connector security maps associated with `connector-Pool1`.

```
asadmin> list-connector-security-maps connector-Pool1
securityMap1
Command list-connector-security-maps executed successfully.
```

### Example 12-12 Listing Principals for a Specific Security Map for a Connector Connection Pool

This example lists the principals associated with `securityMap1`.

```
asadmin> list-connector-security-maps --securitymap securityMap1 connector-Pool1
principal1
principal1
Command list-connector-security-maps executed successfully.
```

### Example 12-13 Listing Principals of All Connector Security Maps for a Connector Connection Pool

This example lists the connector security maps associated with `connector-Pool1`.

```
asadmin> list-connector-security-maps --verbose connector-Pool1
securityMap1
principal1
principal1
Command list-connector-security-maps executed successfully.
```

## To Update a Connector Security Map

Use the `update-connector-security-map` subcommand in remote mode to create or modify a security map for the specified connector connection pool.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List existing connector security maps by using the `list-connector-security-maps` subcommand.
3. Modify a security map for a specific connector connection pool by using the `update-connector-security-map` subcommand.
4. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 12-14 Updating a Connector Security Map

This example adds principals to `securityMap1`.

```
asadmin> update-connector-security-map --poolname connector-pool1  
--addprincipals principal1, principal2 securityMap1  
Command update-connector-security-map executed successfully.
```

## To Delete a Connector Security Map

Use the `delete-connector-security-map` subcommand in remote mode to delete a security map for the specified connector connection pool.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List existing connector connection pools by using the `list-connector-connection-pools` subcommand.
3. Delete a security map for a specific connector connection pool by using the `delete-connector-security-map` subcommand.

Information about options for this subcommand is included in this help page.

### Example 12-15 Deleting a Connector Security Map

This example deletes `securityMap1` from `connector-pool1`.

```
asadmin> delete-connector-security-map --poolname connector-pool1 securityMap1  
Command delete-connector-security-map executed successfully
```

# Administering Connector Work Security Maps

The EIS is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application. The connector work security map is used to map the EIS credentials to the credentials of Eclipse GlassFish security domain.

A security map applies to a particular connector connection pool. One or more named security maps can be associated with a connector connection pool.

The following topics are addressed here:

- [To Create a Connector Work Security Map](#)
- [To List Connector Work Security Maps](#)
- [To Update a Connector Work Security Map](#)
- [To Delete a Connector Work Security Map](#)

## To Create a Connector Work Security Map

Use the `create-connector-work-security-map` subcommand in remote mode to map the caller identity of the work submitted by the connector module (resource adapter) EIS principal or EIS user group to a suitable principal or user group in the Eclipse GlassFish security domain. One or more work security maps can be associated with a connector module.

The connector security map configuration supports the use of the wild card asterisk (\*) to indicate all users or all user groups.

### Before You Begin

Before creating a connector work security map, you must first create a connector connection pool. For instructions, see [To Create a Connector Connection Pool](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create the connector work security map by using the `create-connector-work-security-map` subcommand.

Information about properties for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 12-16 Creating Connector Work Security Maps

The following examples create `workSecurityMap1` and `workSecurityMap2` for `my-resource-adapter-name`.

```
asadmin> create-connector-work-security-map --rname my-resource-adapter-name
```

```
--principalsmap eis-principal-1=server-principal-1,eis-principal-2=server-principal-2,  
eis-principal-3=server-principal-1 workSecurityMap1  
  
asadmin> create-connector-work-security-map --rname my-resource-adapter-name  
--groupsmap eis-group-1=server-group-1,eis-group-2=server-group-2,  
eis-group-3=server-group-1 workSecurityMap2  
Command create-connector-work-security-map executed successfully
```

## See Also

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

## To List Connector Work Security Maps

Use the `list-connector-work-security-maps` subcommand in remote mode to list the work security maps that belong to a specific connector module.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector work security maps by using the `list-connector-work-security-maps` subcommand.

### Example 12-17 Listing the Connector Work Security Maps

This example lists the generic work security maps.

```
asadmin> list-connector-work-security-maps generic-ra  
generic-ra-groups-map: EIS group=eis-group, mapped group=glassfish-group  
generic-ra-principals-map: EIS principal=eis-bar, mapped principal=bar  
generic-ra-principals-map: EIS principal=eis-foo, mapped principal=foo  
Command list-connector-work-security-maps executed successfully.
```

## See Also

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

## To Update a Connector Work Security Map

Use the `update-connector-work-security-map` subcommand in remote to modify a work security map that belongs to a specific resource adapter (connector module).

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector work security maps by using the `list-connector-work-security-maps`

subcommand.

3. If necessary, notify users that the connector work security map is being modified.
4. Update a connector work security map by using the `update-connector-work-security-map` subcommand.

#### Example 12-18 Updating a Connector Work Security Map

This example removes a principal from a work security map.

```
asadmin> update-connector-work-security-map --raname generic-ra  
--removeprincipals eis-foo generic-ra-principals-map  
Command update-connector-work-security-map executed successfully.
```

#### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help update-connector-work-security-map` at the command line.

## To Delete a Connector Work Security Map

Use the `delete-connector-work-security-map` subcommand in remote mode to delete a work security map that belongs to a specific connector module (resource adapter).

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the connector work security maps by using the `list-connector-work-security-maps` subcommand.
3. Delete a connector work security map by using the `delete-connector-work-security-map` subcommand.

#### Example 12-19 Deleting a Connector Work Security Map

This example deletes the `worksecuritymap1` map from the `my_ra` connector module.

```
asadmin> delete-connector-work-security-map --raname my_ra worksecuritymap1  
Command delete-connector-work-security-map executed successfully.
```

#### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-connector-work-security-map` at the command line.

# Administering Administered Objects

Packaged within a connector module, an administered object provides specialized functionality for an application. For example, an administered object might provide access to a parser that is specific to the connector module and its associated EIS.

The following topics are addressed here:

- [To Create an Administered Object](#)
- [To List Administered Objects](#)
- [To Update an Administered Object](#)
- [To Delete an Administered Object](#)

## To Create an Administered Object

Use the `create-admin-object` subcommand to create an administered object resource. When creating an administered object resource, name-value pairs are created, and the object is associated to a JNDI name.

### Before You Begin

The resource adapter must be deployed before running this subcommand (`jmsrar.rar`).

1. Create an administered object by using the `create-admin-object` subcommand.

Information about properties for the subcommand is included in this help page.

2. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 12-20 Creating an Administered Object

For this example, the `jakarta.jms.Queue` resource type is obtained from the `ra.xml` file. The JNDI name of the new administered object is `jms/samplequeue`.

```
asadmin> create-admin-object --restype jakarta.jms.Queue --raname jmsra
--description "sample administered object" --property Name=sample_jmsqueue
jms/samplequeue
Command create-admin-object executed successfully
```

### See Also

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

## To List Administered Objects

Use the `list-admin-object` subcommand in remote mode to list the existing administered objects.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the administered objects by using the `list-admin-objects` subcommand.

### Example 12-21 Listing Administered Objects

This example lists the existing administered objects.

```
asadmin> list-admin-objects  
jms/samplequeue  
Command list-admin-objects executed successfully
```

### See Also

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

## To Update an Administered Object

Use the `get` and `set` subcommands to view and change the values of the administered objects properties.

1. List the administered objects by using the `list-admin-objects` subcommand.
2. View the properties of the administered object by using the `get` subcommand. For example:

```
asadmin> get domain.resources.admin-object-resource.jms/samplequeue.*
```

3. Set the property of the administered object by using the `set` subcommand. For example:

```
asadmin> set domain.resources.admin-object-resource.jms/samplequeue.enabled=false
```

4. If needed, restart the server. Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

## To Delete an Administered Object

Use the `delete-admin-object` subcommand to delete an administered objects.

1. List the administered objects by using the `list-admin-objects` subcommand.
2. If necessary, notify users that the administered object is being deleted.
3. Delete an administered object by using the `delete-admin-object` subcommand.

#### Example 12-22 Deleting an Administered Object

This example deletes the administered object with the JNDI name `jms/samplequeue`.

```
asadmin> delete-admin-object jms/samplequeue
Command delete-admin-object executed successfully
```

#### See Also

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

# 13 Administering Internet Connectivity

This chapter provides procedures for performing internet connectivity tasks in the Eclipse GlassFish 7 environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Internet Connectivity](#)
- [Administering HTTP Network Listeners](#)
- [Administering Virtual Servers](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About Internet Connectivity

The HTTP service provides functionality for deploying web applications and for making deployed web applications accessible by Internet clients, either in a single application server instance or in a cluster of multiple server instances. HTTP services are provided by two kinds of related objects: listeners and virtual servers.

For more information about clusters, see the [Eclipse GlassFish High Availability Administration Guide](#).

The following topics are addressed here:

- [About HTTP Network Listeners](#)
- [About Virtual Servers](#)

## About HTTP Network Listeners

An HTTP listener, also known as a network listener, is a listen socket that has an Internet Protocol (IP) address, a port number, a server name, and a default virtual server. Each virtual server provides connections between the server and clients through one or more listeners. Each listener must have a unique combination of port number and IP address. For example, an HTTP listener can listen for a host on all configured IP addresses on a given port by specifying the IP address 0.0.0.0. Alternatively, the listener can specify a unique IP address for each listener while using the same port.

Because an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers, or with different IP addresses and the same port number (if your host was configured to respond to these addresses). However, if an HTTP listener uses the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot create HTTP listeners for additional IP addresses that listen on the same port for a specific IP

address. For example, if an HTTP listener uses 0.0.0.0:8080 (all IP addresses on port 8080), another HTTP listener cannot use 1.2.3.4:8080. The host running the Eclipse GlassFish typically has access to only one IP address. HTTP listeners typically use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. However, if the host does have access to more than one IP address, each address can serve a different purpose.

To access a web application deployed on Eclipse GlassFish, use the URL <http://localhost:8080/> (or <https://localhost:8081/> for a secure application), along with the context root specified for the web application.

To access the Administration Console, use the URL <https://localhost:4848/> or <http://localhost:4848/asadmin/> (console default context root).

## About Virtual Servers

A virtual server, sometimes called a virtual host, is an object that allows the same physical server to host multiple Internet domain names. All virtual servers hosted on the same physical server share the IP address of that physical server. A virtual server associates a domain name for a server (such as [www.aaa.com](http://www.aaa.com)) with the particular server on which Eclipse GlassFish is running. Each virtual server must be registered with the DNS server for your network.



Do not confuse an Internet domain with the administrative domain of Eclipse GlassFish.

For example, assume that you want to host the following domains on your physical server: [www.aaa.com](http://www.aaa.com), [www.bbb.com](http://www.bbb.com), and [www.ccc.com](http://www.ccc.com). Assume that these domains are respectively associated with web modules [web1](#), [web2](#), and [web3](#). This means that the following URLs are handled by your physical server:

```
http://www.aaa.com:8080/web1  
http://www.bbb.com:8080/web2  
http://www.ccc.com:8080/web3
```

The first URL is mapped to virtual server [www.aaa.com](http://www.aaa.com), the second URL is mapped to virtual server [www.bbb.com](http://www.bbb.com), and the third is mapped to virtual server [www.ccc.com](http://www.ccc.com). For this mapping to work, [www.aaa.com](http://www.aaa.com), [www.bbb.com](http://www.bbb.com), and [www.ccc.com](http://www.ccc.com) must all resolve to your physical server's IP address and each virtual server must be registered with the DNS server for your network. In addition, on a UNIX system, add these domains to your `/etc/hosts` file (if the setting for `hosts` in your `/etc/nsswitch.conf` file includes `files`).

## Administering HTTP Network Listeners

By default, when Eclipse GlassFish starts, the following HTTP listeners are started automatically:

- HTTP listeners associated with the virtual server named `server`:
  - The listener named `http-listener-1` does not have security enabled.
  - The listener named `http-listener-2` has security enabled.
- An HTTP listener named `admin-listener`, associated with the virtual server named `_asadmin`. For this listener, security is not enabled.

The following table describes the Eclipse GlassFish default ports for the listeners that use ports.

Table 13-1 Default Ports for Listeners

Listener	Default Port	Description
Administrative server	4848	A domain's administrative server is accessed by the Administration Console and the <code>asadmin</code> utility. For the Administration Console, specify the port number in the URL of the browser. When running an <code>asadmin</code> subcommand remotely, specify the port number by using the <code>--port</code> option.
HTTP	8080	The web server listens for HTTP requests on a port. To access deployed web applications and services, clients connect to this port.
HTTPS	8181	Web applications configured for secure communications listen on a separate port.

The following topics are addressed here:

- [To Create an Internet Connection](#)
- [Administering HTTP Protocols](#)
- [Administering HTTP Configurations](#)
- [Administering HTTP Transports](#)
- [Administering HTTP Network Listeners](#)

## To Create an Internet Connection

Use the subcommands in this procedure to create an internet connection with the full range of listener options. A network listener is created behind the scenes. For the shortcut version of this process , see [To Create an HTTP Network Listener](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create an HTTP or HTTPS protocol by using the `create-protocol` subcommand with the `--securityenabled` option. To use the built-in `http-listener-1` HTTP protocol, or `http-listener-2` HTTPS protocol, skip this step.
3. Create an HTTP configuration by using the `create-http` subcommand. To use a built-in protocol,

- skip this step.
4. Create a transport by using the `create-transport` subcommand. To use the built-in `tcp` transport, skip this step.
  5. Create a thread pool by using the `create-threadpool` subcommand. To avoid using a thread pool, or to use the built-in `http-thread-pool` thread pool, skip this step. For additional thread pool information, see [Administering Thread Pools](#).
  6. Create an HTTP listener by using the `create-network-listener` subcommand. Specify a protocol and transport, optionally a thread pool.
  7. To apply your changes, restart Eclipse GlassFish. See [To Restart a Domain](#).

#### See Also

You can also view the full syntax and options of the subcommand by typing a command such as `asadmin help create-http-listener` at the command line.

## Administering HTTP Protocols

Each HTTP listener has an HTTP protocol, which is created either by using the `create-protocol` subcommand or by using the built-in protocols that are applied when you follow the instructions in [To Create an HTTP Network Listener](#).

The following topics are addressed here:

- [To Create a Protocol](#)
- [To List Protocols](#)
- [To Delete a Protocol](#)

### To Create a Protocol

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

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a protocol by using the `create-protocol`

Information about options and properties for the subcommand are included in this help page.

#### Example 13-1 Creating an HTTP Protocol

This example creates a protocol named `http-1` with security enabled.

```
asadmin> create-protocol --securityenabled=true http-1
Command create-protocol executed successfully.
```

## See Also

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

## To List Protocols

Use the `list-protocols` subcommand in remote mode to list the existing HTTP protocols.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing protocols by using the `list-protocols` subcommand.

### Example 13-2 Listing the Protocols

This example lists the existing protocols.

```
asadmin> list-protocols
admin-listener
http-1
http-listener-1
http-listener-2
Command list-protocols executed successfully.
```

## See Also

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

## To Delete a Protocol

Use the `delete-protocol` subcommand in remote mode to remove a protocol.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Delete a protocol by using the `delete-protocol` subcommand

### Example 13-3 Deleting a Protocol

This example deletes the protocol named `http-1`.

```
asadmin> delete-protocol http-1
Command delete-protocol executed successfully.
```

## See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-`

**protocol** at the command line.

## Administering HTTP Configurations

Each HTTP listener has an HTTP configuration, which is created either by using the **create-http** subcommand or by using the built-in configurations that are applied when you follow the instructions in [To Create an HTTP Network Listener](#).

The following topics are addressed here:

- [To Create an HTTP Configuration](#)
- [To Delete an HTTP Configuration](#)

### To Create an HTTP Configuration

Use the **create-http** subcommand in remote mode to create a set of HTTP parameters for a protocol. This set of parameters configures one or more network listeners,

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create an HTTP configuration by using the **create-http** subcommand. Information about options and properties for the subcommand are included in this help page.

#### Example 13-4 Creating an HTTP Configuration

This example creates an HTTP parameter set for the protocol named **http-1**.

```
asadmin> create-http --timeout-seconds 60 --default-virtual-server server http-1
Command create-http executed successfully.
```

#### See Also

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

### To Delete an HTTP Configuration

Use the **delete-http** subcommand in remote mode to remove HTTP parameters from a protocol.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Delete the HTTP parameters from a protocol by using the **delete-http** subcommand.

#### Example 13-5 Deleting an HTTP Configuration

This example deletes the HTTP parameter set from a protocol named `http-1`.

```
asadmin> delete-http http-1
Command delete-http executed successfully.
```

## See Also

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

## Administering HTTP Transports

Each HTTP listener has an HTTP transport, which is created either by using the `create-transport` subcommand or by using the built-in transports that are applied when you follow the instructions in [To Create an HTTP Network Listener](#).

The following topics are addressed here:

- [To Create a Transport](#)
- [To List Transports](#)
- [To Delete a Transport](#)

### To Create a Transport

Use the `create-transport` subcommand in remote mode to create a transport for a network listener,

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a transport by using the `create-transport` subcommand. Information about options and properties for the subcommand are included in this help page.

#### Example 13-6 Creating a Transport

This example creates a transport named `http1-trans` that uses a non-default number of acceptor threads.

```
asadmin> create-transport --acceptorthreads 100 http1-trans
Command create-transport executed successfully.
```

## See Also

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

## To List Transports

Use the `list-transports` subcommand in remote mode to list the existing HTTP transports.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing transports by using the `list-transports` subcommand.

### Example 13-7 Listing HTTP Transports

This example lists the existing transports.

```
asadmin> list-transports
http1-trans
tcp
Command list-transports executed successfully.
```

### See Also

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

## To Delete a Transport

Use the `delete-transport` subcommand in remote mode to remove a transport.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Delete a transport by using the `delete-transport` subcommand.

### Example 13-8 Deleting a Transport

This example deletes the transport named `http1-trans`.

```
asadmin> delete-transport http1-trans
Command delete-transport executed successfully.
```

### See Also

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

## Administering HTTP Network Listeners

The following topics are addressed here:

- [To Create an HTTP Network Listener](#)

- [To List HTTP Network Listeners](#)
- [To Update an HTTP Network Listener](#)
- [To Delete an HTTP Network Listener](#)
- [To Configure an HTTP Listener for SSL](#)
- [To Configure Optional Client Authentication for SSL](#)
- [To Configure a Custom SSL Implementation](#)
- [To Delete SSL From an HTTP Listener](#)
- [To Assign a Default Virtual Server to an HTTP Listener](#)

## To Create an HTTP Network Listener

Use the `create-http-listener` subcommand or the `create-network-listener` subcommand in remote mode to create a listener. These subcommands provide backward compatibility and also provide a shortcut for creating network listeners that use the HTTP protocol. Behind the scenes, a network listener is created as well as its associated protocol, transport, and HTTP configuration. This method is a convenient shortcut, but it gives access to only a limited number of options. If you want to specify the full range of listener options, follow the instructions in [To Create an Internet Connection](#).

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create an HTTP network listener by using the `create-network-listener` subcommand or the `create-http-listener` subcommand.
3. If needed, restart the server.

If you edit the special HTTP network listener named `admin-listener`, you must restart the server for changes to take effect. See [To Restart a Domain](#).

### Example 13-9 Creating an HTTP Listener

This example creates an HTTP listener named `sampleListener` that uses a non-default number of acceptor threads. Security is not enabled at runtime.

```
asadmin> create-http-listener --listeneraddress 0.0.0.0
--listenerport 7272 --defaultvs server --servername host1.sun.com
--acceptorthreads 100 --securityenabled=false
--enabled=false sampleListener
Command create-http-listener executed successfully.
```

### Example 13-10 Creating a Network Listener

This example creates a network listener named `sampleListener` that is not enabled at runtime:

```
asadmin> create-network-listener --listenerport 7272 protocol http-1
```

```
--enabled=false sampleListener  
Command create-network-listener executed successfully.
```

## See Also

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

## To List HTTP Network Listeners

Use the `list-http-listeners` subcommand or the `list-network-listeners` subcommand in remote mode to list the existing HTTP listeners.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List HTTP listeners by using the `list-http-listeners` or `list-network-listeners` subcommand.

### Example 13-11 Listing HTTP Listeners

This example lists the HTTP listeners. The same output is given if you use the `list-network-listeners` subcommand.

```
asadmin> list-http-listeners  
admin-listener  
http-listener-2  
http-listener-1  
Command list-http-listeners executed successfully.
```

## See Also

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

## To Update an HTTP Network Listener

1. List HTTP listeners by using the `list-http-listeners` or `list-network-listeners` subcommand.
2. Modify the values for the specified listener by using the `set` subcommand.

The listener is identified by its dotted name.

### Example 13-12 Updating an HTTP Network Listener

This example changes `security-enabled` to `false` on `http-listener-2`.

```
asadmin> set server.network-config.protocols.protocol.http-listener-2.security-  
enabled=false
```

Command set executed successfully.

## To Delete an HTTP Network Listener

Use the `delete-http-listener` subcommand or the `delete-network-listener` subcommand in remote mode to delete an existing HTTP listener. This disables secure communications for the listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List HTTP listeners by using the `list-http-listeners` subcommand.
3. Delete an HTTP listener by using the `delete-http-listener` or `delete-network-listener` subcommand.
4. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 13-13 Deleting an HTTP Listener

This example deletes the HTTP listener named `sampleListener`:

```
asadmin> delete-http-listener sampleListener  
Command delete-http-listener executed successfully.
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-http-listener` or `asadmin help delete-network-listener` at the command line.

## To Configure an HTTP Listener for SSL

Use the `create-ssl` subcommand in remote mode to create and configure an SSL element in the specified listener. This enables secure communication for the listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Configure an HTTP listener by using the `create-ssl` subcommand.
3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 13-14 Configuring an HTTP Listener for SSL

This example enables the HTTP listener named `http-listener-1` for SSL:

```
asadmin> create-ssl --type http-listener --certname sampleCert http-listener-1
```

```
Command create-ssl executed successfully.
```

## See Also

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

## To Configure Optional Client Authentication for SSL

In Eclipse GlassFish, you can configure the SSL protocol of an HTTP listener such that it requests a certificate before permitting a client connection, but does not refuse a connection if the client does not provide one. To enable this feature, set the `client-auth` property of the SSL protocol to the value `want`. For example:

```
asadmin> set configs.config.config-name.network-config.protocols.\
protocol.listener-name.ssl.client-auth=want
```

## To Configure a Custom SSL Implementation

In Eclipse GlassFish, you can configure the SSL protocol of an HTTP listener such that it uses a custom implementation of SSL. To enable this feature, set the `classname` property of the SSL protocol to the name of a class that implements the `com.sun.grizzly.util.net.SSLImplementation` interface. For example:

```
asadmin> set configs.config.config-name.network-config.protocols.\
protocol.listener-name.ssl.classname=SSLImplementation-class-name
```

By default, Eclipse GlassFish uses the implementation `com.sun.enterprise.security.ssl.GlassfishSSLImpl` for the SSL protocol.

## To Delete SSL From an HTTP Listener

Use the `delete-ssl` subcommand in remote mode to delete the SSL element in the specified listener. This disables secure communications for the listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Delete SSL from an HTTP listener by using the `delete-ssl` subcommand.
3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

## Example 13-15 Deleting SSL From an HTTP Listener

This example disables SSL for the HTTP listener named `http-listener-1`:

```
asadmin> delete-ssl --type http-listener http-listener-1  
Command delete-http-listener executed successfully.
```

### See Also

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

## To Assign a Default Virtual Server to an HTTP Listener

1. In the Administration Console, open the HTTP Service component under the relevant configuration.
2. Open the HTTP Listeners component under the HTTP Service component.
3. Select or create a new HTTP listener.
4. Select from the Default Virtual Server drop-down list.

For more information, see [To Assign a Default Web Module to a Virtual Server](#).

### See Also

For details, click the Help button in the Administration Console from the HTTP Listeners page.

# Administering Virtual Servers

A virtual server is a virtual web server that serves content targeted for a specific URL. Multiple virtual servers can serve content using the same or different host names, port numbers, or IP addresses. The HTTP service directs incoming web requests to different virtual servers based on the URL.

When you first install Eclipse GlassFish, a default virtual server is created. You can assign a default virtual server to each new HTTP listener you create.

Web applications and Jakarta EE applications containing web components (web modules) can be assigned to virtual servers during deployment. A web module can be assigned to more than one virtual server, and a virtual server can have more than one web module assigned to it. If you deploy a web application and don't specify any assigned virtual servers, the web application is assigned to all currently defined virtual servers. If you then create additional virtual servers and want to assign existing web applications to them, you must redeploy the web applications. For more information about deployment, see the [Eclipse GlassFish Application Deployment Guide](#).

You can define virtual server properties using the `asadmin set` command. For example:

```
asadmin> set server-config.http-service.virtual-server.MyVS.property.sso-enabled="true"
```

Some virtual server properties can be set for a specific web application. For details, see "[glassfish-web-app](#)" in Eclipse GlassFish Application Deployment Guide.

The following topics are addressed here:

- [To Create a Virtual Server](#)
- [To List Virtual Servers](#)
- [To Update a Virtual Server](#)
- [To Delete a Virtual Server](#)
- [To Assign a Default Web Module to a Virtual Server](#)
- [To Assign a Virtual Server to an Application or Module](#)
- [To Set JSESSIONIDSSO Cookie Attributes](#)

## To Create a Virtual Server

By default, when Eclipse GlassFish starts, the following virtual servers are started automatically:

- A virtual server named `server`, which hosts all user-defined web modules.

For development, testing, and deployment of web services in a non-production environment, `server` is often the only virtual server required.

- A virtual server named `_asadmin`, which hosts all administration-related web modules (specifically, the Administration Console). This server is restricted, which means that you cannot deploy web modules to this virtual server.

In a production environment, additional virtual servers provide hosting facilities for users and customers so that each appears to have its own web server, even though there is only one physical server.

Use the `create-virtual-server` subcommand in remote mode to create the named virtual server.

### Before You Begin

A virtual server must specify an existing HTTP listener. Because the virtual server cannot specify an HTTP listener that is already being used by another virtual server, create at least one HTTP listener before creating a new virtual server.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a virtual server by using the `create-virtual-server` subcommand.

Information about properties for this subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 13-16 Creating a Virtual Server

This example creates a virtual server named `sampleServer` on `localhost`.

```
asadmin> create-virtual-server sampleServer
Command create-virtual-server executed successfully.
```

#### See Also

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

## To List Virtual Servers

Use the `list-virtual-servers` subcommand in remote mode to list the existing virtual servers.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List virtual servers by using the `list-virtual-servers` subcommand.

#### Example 13-17 Listing Virtual Servers

This example lists the virtual servers for `localhost`.

```
asadmin> list-virtual-servers
sampleListener
admin-listener
http-listener-2
http-listener-1
Command list-http-listeners executed successfully.
```

#### See Also

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

## To Update a Virtual Server

1. List virtual servers by using the `list-virtual-servers` subcommand.

2. Modify the values for the specified virtual server by using the `set` subcommand.

The virtual server is identified by its dotted name.

## To Delete a Virtual Server

Use the `delete-virtual-server` subcommand in remote mode to delete an existing virtual server.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List virtual servers by using the `list-virtual-servers` subcommand.
3. If necessary, notify users that the virtual server is being deleted.
4. Delete a virtual server by using the `delete-virtual-server` subcommand.
5. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 13-18 Deleting a Virtual Server

This example deletes the virtual server named `sampleServer` from `localhost`.

```
asadmin> delete-virtual-server sampleServer
Command delete-virtual-server executed successfully.
```

### See Also

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

## To Assign a Default Web Module to a Virtual Server

A default web module can be assigned to the default virtual server and to each new virtual server. To access the default web module for a virtual server, point the browser to the URL for the virtual server, but do not supply a context root. For example:

```
http://myvserver:3184/
```

A virtual server with no default web module assigned serves HTML or JavaServer Pages ( JSP) content from its document root, which is usually domain-dir/`docroot`. To access this HTML or JSP content, point your browser to the URL for the virtual server, do not supply a context root, but specify the target file.

For example:

```
http://myvserver:3184/hellothere.jsp
```

## To Assign a Virtual Server to an Application or Module

You can assign a virtual server to a deployed application or web module.

### Before You Begin

The application or module must already be deployed. For more information, see the [Eclipse GlassFish Application Deployment Guide](#).

1. In the Administration Console, open the HTTP Service component under the relevant configuration.
2. Open the Virtual Servers component under the HTTP Service component.
3. Select the virtual server to which you want to assign a default web module.
4. Select the application or web module from the Default Web Module drop-down list.

For more information, see [To Assign a Default Web Module to a Virtual Server](#).

## To Set JSESSIONIDSSO Cookie Attributes

Use the `sso-cookie-http-only` and `sso-cookie-secure` virtual server attributes to set the `HttpOnly` and `Secure` attributes of any `JSESSIONIDSSO` cookies associated with web applications deployed to the virtual server.

Use the `set` subcommand to set the value of the `sso-cookie-http-only` and `sso-cookie-secure` virtual server attributes.

The values supported for these attributes are as follows:

### `sso-cookie-http-only`

A boolean value that specifies whether the `HttpOnly` attribute is included in `JSESSIONIDSSO` cookies. When set to `true`, which is the default, the `HttpOnly` attribute is included. When set to `false`, the `HttpOnly` attribute is not included.

### `sso-cookie-secure`

A string value that specifies whether the `Secure` attribute is included in `JSESSIONIDSSO` cookies. Allowed values are as follows:

- `true` — The `Secure` attribute is included.
- `false` — The `Secure` attribute is not included.
- `dynamic` — The `Secure` attribute setting is inherited from the first session participating in SSO. This is the default value.



# 14 Administering Concurrent Resources

This chapter provides procedures for administering concurrent resources in the Eclipse GlassFish environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- [About Concurrent Resources](#)
- [Default Concurrent Resources](#)
- [Configuring Context Services](#)
- [Configuring Managed Thread Factories](#)
- [Configuring Managed Executor Services](#)
- [Configuring Managed Scheduled Executor Services](#)

Instructions for accomplishing these tasks by using the Administration Console are contained in the Administration Console online help.

## About Concurrent Resources

Concurrent resources are managed objects that provide concurrency capabilities to Jakarta EE applications. In Eclipse GlassFish, you configure concurrent resources and make them available for use by application components such as servlets and EJBs. Concurrent resources are accessed through JNDI lookup or resource injection.

Concurrent resources are resources of the following types:

- Context services. See [Configuring Context Services](#).
- Managed thread factories. See [Configuring Managed Thread Factories](#).
- Managed executor services. See [Configuring Managed Executor Services](#).
- Managed scheduled executor services. See [Configuring Managed Scheduled Executor Services](#).

For detailed information about concurrent resources, see [Concurrency Utilities](#) in The Jakarta EE Tutorial. Also see [Java Specification Request 236: Concurrency Utilities for Jakarta EE](#).

## Default Concurrent Resources

When you create a concurrent resource, you specify a unique JNDI name for the resource. Applications use this name to access the resource.

The Jakarta EE standard specifies that certain default resources be made available to applications, and defines specific JNDI names for these default resources. Eclipse GlassFish makes these names

available through the use of logical JNDI names, which map Jakarta EE standard JNDI names to specific Eclipse GlassFish resources. For concurrent resources, the mappings are as follows:

#### **java:comp/DefaultContextService**

This Jakarta EE standard name is mapped to the `concurrent/_defaultContextService` resource.

#### **java:comp/DefaultManagedThreadFactory**

This Jakarta EE standard name is mapped to the `concurrent/_defaultManagedThreadFactory` resource.

#### **java:comp/DefaultManagedExecutorService**

This Jakarta EE standard name is mapped to the `concurrent/_defaultManagedExecutorService` resource.

#### **java:comp/DefaultManagedScheduledExecutorService**

This Jakarta EE standard name is mapped to the `concurrent/_defaultManagedScheduledExecutorService` resource.

## **Configuring Context Services**

Context services are used to create dynamic proxy objects that capture the context of a container and enable applications to run within that context at a later time. The context of the container is propagated to the thread executing the task.

The following tasks are used to administer context service resources:

- [To Create a Context Service](#)
- [To List Context Services](#)
- [To Update a Context Service](#)
- [To Delete a Context Service](#)

### **To Create a Context Service**

Use the `create-context-service` subcommand in remote mode to create a context service resource.

Because all JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a context service, use only the `concurrent/'name` format. For example, `'concurrent/Context1`.

For more information about the default context service resource included with Eclipse GlassFish, see [Default Concurrent Resources](#).



Creating a context service resource is a dynamic event and typically does not require server restart. Applications can use a resource as soon as it is created. However, if an application tried to use a resource before it was created, and that

resource is created later, the application or the server must be restarted. Otherwise, the application will not be able to locate the resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a context service by using the `create-context-service` subcommand.
3. If necessary, notify users that the new resource has been created.

#### Example 14-1 Creating a Context Service

This example creates a context service resource named `concurrent/Context1`.

```
asadmin> create-context-service concurrent/Context1
Context service concurrent/Context1 created successfully.
Command create-context-service executed successfully.
```

#### See Also

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

## To List Context Services

Use the `list-context-services` subcommand in remote mode to list the existing context service resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List context service resources by using the `list-context-services` subcommand.

#### Example 14-2 Listing Context Services

This example lists context service resources on the default server instance, `server`.

```
asadmin> list-context-services
concurrent/_defaultContextService
concurrent/Context1
concurrent/Context2
Command list-context-services executed successfully.
```

#### See Also

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

## To Update a Context Service

You can change all of the settings for an existing context service resource except its JNDI name. Use the `get` and `set` subcommands to view and change the values of the context service attributes.



When a resource is updated, the existing resource is shut down and recreated. If an application used the resource prior to the update, the application or the server must be restarted.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the context service resources by using the `list-context-services` subcommand.
3. View the attributes of a specific context service by using the `get` subcommand. For example:

```
asadmin> get resources.context-service.concurrent/Context1.*
```

4. Set an attribute of the context service by using the `set` subcommand. For example:

```
asadmin> set resources.context-service.concurrent/Context1.deployment-order=120
```

## To Delete a Context Service

Use the `delete-context-service` subcommand in remote mode to delete an existing context service. Deleting a context service is a dynamic event and does not require server restart.

Before deleting a context service resource, all associations to the resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the context service resources by using the `list-context-services` subcommand.
3. If necessary, notify users that the context service is being deleted.
4. Delete the context service by using the `delete-context-service` subcommand.

### Example 14-3 Deleting a Context Service

This example deletes the context service resource named `concurrent/Context1`.

```
asadmin> delete-context-service concurrent/Context1
Context service concurrent/Context1 deleted successfully.
Command delete-context-service executed successfully.
```

### See Also

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

# Configuring Managed Thread Factories

Managed thread factories are used by applications to create managed threads on demand. The threads are started and managed by the container. The context of the container is propagated to the thread executing the task.

The following tasks are used to administer managed thread factory resources:

- [To Create a Managed Thread Factory](#)
- [To List Managed Thread Factories](#)
- [To Update a Managed Thread Factory](#)
- [To Delete a Managed Thread Factory](#)

## To Create a Managed Thread Factory

Use the `create-managed-thread-factory` subcommand in remote mode to create a managed thread factory resource.

Because all JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a managed thread factory, use only the `concurrent/name` format. For example, `concurrent/Factory1`.

For more information about the default managed thread factory resource included with Eclipse GlassFish, see [Default Concurrent Resources](#).



Creating a managed thread factory resource is a dynamic event and typically does not require server restart. Applications can use a resource as soon as it is created. However, if an application tried to use a resource before it was created, and that resource is created later, the application or the server must be restarted. Otherwise, the application will not be able to locate the resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a managed thread factory by using the `create-managed-thread-factory` subcommand.
3. If necessary, notify users that the new resource has been created.

### Example 14-4 Creating a Managed Thread Factory

This example creates a managed thread factory resource named `concurrent/Factory1`.

```
asadmin> create-managed-thread-factory concurrent/Factory1
Managed thread factory concurrent/Factory1 created successfully.
Command create-managed-thread-factory executed successfully.
```

## See Also

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

## To List Managed Thread Factories

Use the `list-managed-thread-factories` subcommand in remote mode to list the existing managed thread factory resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List managed thread factory resources by using the `list-managed-thread-factories` subcommand.

### Example 14-5 Listing Managed Thread Factories

This example lists managed thread factory resources on the default server instance, `server`.

```
asadmin> list-managed-thread-factories
concurrent/_defaultManagedThreadFactory
concurrent/Factory1
concurrent/Factory2
Command list-managed-thread-factories executed successfully.
```

## See Also

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

## To Update a Managed Thread Factory

You can change all of the settings for an existing managed thread factory resource except its JNDI name. Use the `get` and `set` subcommands to view and change the values of the managed thread factory attributes.

 When a resource is updated, the existing resource is shut down and recreated. If applications used the resource prior to the update, the application or the server must be restarted.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the managed thread factory resources by using the `list-managed-thread-factories` subcommand.
3. View the attributes of a managed thread factory by using the `get` subcommand. For example:

```
asadmin> get resources.managed-thread-factory.concurrent/Factory1.*
```

- Set an attribute of the managed thread factory by using the `set` subcommand. For example:

```
asadmin> set resources.managed-thread-factory.concurrent/Factory1.deployment-order=120
```

## To Delete a Managed Thread Factory

Use the `delete-managed-thread-factory` subcommand in remote mode to delete an existing managed thread factory. Deleting a managed thread factory is a dynamic event and does not require server restart.

Before deleting a managed thread factory resource, all associations to the resource must be removed.

- Ensure that the server is running. Remote subcommands require a running server.
- List the managed thread factory resources by using the `list-managed-thread-factories` subcommand.
- If necessary, notify users that the managed thread factory is being deleted.
- Delete the managed thread factory by using the `delete-managed-thread-factory` subcommand.

### Example 14-6 Deleting a Managed Thread Factory

This example deletes the managed thread factory resource named `concurrent/Factory1`.

```
asadmin> delete-managed-thread-factory concurrent/Factory1
Managed thread factory concurrent/Factory1 deleted successfully.
Command delete-managed-thread-factory executed successfully.
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-managed-thread-factory` at the command line.

## Configuring Managed Executor Services

Managed executor services are used by applications to execute submitted tasks asynchronously. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task.

The following tasks are used to administer managed executor service resources:

- [To Create a Managed Executor Service](#)
- [To List Managed Executor Services](#)
- [To Update a Managed Executor Service](#)
- [To Delete a Managed Executor Service](#)

## To Create a Managed Executor Service

Use the `create-managed-executor-service` subcommand in remote mode to create a managed executor service resource.

Because all JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a managed executor service, use only the `concurrent/'name'` format. For example, `'concurrent/Executor1'`.

For more information about the default managed executor service resource included with Eclipse GlassFish, see [Default Concurrent Resources](#).



Creating a managed executor service resource is a dynamic event and typically does not require server restart. Applications can use a resource as soon as it is created. However, if an application tried to use a resource before it was created, and that resource is created later, the application or the server must be restarted. Otherwise, the application will not be able to locate the resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a managed executor service by using the `create-managed-executor-service` subcommand.
3. If necessary, notify users that the new resource has been created.

### Example 14-7 Creating a Managed Executor Service

This example creates a managed executor service resource named `concurrent/Executor1`.

```
asadmin> create-managed-executor-service concurrent/Executor1
Managed executor service concurrent/Executor1 created successfully.
Command create-managed-executor-service executed successfully.
```

### See Also

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

## To List Managed Executor Services

Use the `list-managed-executor-services` subcommand in remote mode to list the existing managed executor service resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List managed executor service resources by using the `list-managed-executor-services` subcommand.

### Example 14-8 Listing Managed Executor Services

This example lists managed executor service resources on the default server instance, `server`.

```
asadmin> list-managed-executor-services
concurrent/_defaultManagedExecutorService
concurrent/Executor1
concurrent/Executor2
Command list-managed-executor-services executed successfully.
```

### See Also

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

## To Update a Managed Executor Service

You can change all of the settings for an existing managed executor service resource except its JNDI name. Use the `get` and `set` subcommands to view and change the values of the managed executor service attributes.



When a resource is updated, the existing resource is shut down and recreated. If applications used the resource prior to the update, the application or the server must be restarted.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the managed executor service resources by using the `list-managed-executor-services` subcommand.
3. View the attributes of a managed executor service by using the `get` subcommand. For example:

```
asadmin> get resources.managed-executor-service.concurrent/Executor1.*
```

4. Set an attribute of the managed executor service by using the `set` subcommand. For example:

```
asadmin> set resources.managed-executor-service.concurrent/Executor1.deployment-
```

## To Delete a Managed Executor Service

Use the `delete-managed-executor-service` subcommand in remote mode to delete an existing managed executor service. Deleting a managed executor service is a dynamic event and does not require server restart.

Before deleting a managed executor service resource, all associations to the resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the managed executor service resources by using the `list-managed-executor-services` subcommand.
3. If necessary, notify users that the managed executor service is being deleted.
4. Delete the managed executor service by using the `delete-managed-executor-service` subcommand.

### Example 14-9 Deleting a Managed Executor Service

This example deletes the managed executor service resource named `concurrent/Executor1`.

```
asadmin> delete-managed-executor-service concurrent/Executor1
Managed executor service concurrent/Executor1 deleted successfully.
Command delete-managed-executor-service executed successfully.
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-managed-executor-service` at the command line.

## Configuring Managed Scheduled Executor Services

Managed scheduled executor services are used by applications to execute submitted tasks asynchronously at specific times. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task.

The following tasks are used to administer managed scheduled executor service resources:

- [To Create a Managed Scheduled Executor Service](#)
- [To List Managed Scheduled Executor Services](#)
- [To Update a Managed Scheduled Executor Service](#)

- [To Delete a Managed Scheduled Executor Service](#)

## To Create a Managed Scheduled Executor Service

Use the `create-managed-scheduled-executor-service` subcommand in remote mode to create a managed scheduled executor service resource.

Because all JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a managed scheduled executor service, use only the `concurrent/'name` format. For example, `'concurrent/ScheduledExecutor1`.

For more information about the default managed scheduled executor service resource included with Eclipse GlassFish, see [Default Concurrent Resources](#).



Creating a managed scheduled executor service resource is a dynamic event and typically does not require server restart. Applications can use a resource as soon as it is created. However, if an application tried to use a resource before it was created, and that resource is created later, the application or the server must be restarted. Otherwise, the application will not be able to locate the resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a managed scheduled executor service by using the `create-managed-scheduled-executor-service` subcommand.
3. If necessary, notify users that the new resource has been created.

### Example 14-10 Creating a Managed Scheduled Executor Service

This example creates a managed scheduled executor service resource named `concurrent/ScheduledExecutor1`.

```
asadmin> create-managed-scheduled-executor-service concurrent/ScheduledExecutor1
Managed scheduled executor service concurrent/ScheduledExecutor1 created successfully.
Command create-managed-scheduled-executor-service executed successfully.
```

### See Also

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

## To List Managed Scheduled Executor Services

Use the `list-managed-scheduled-executor-services` subcommand in remote mode to list the existing managed scheduled executor service resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List managed scheduled executor service resources by using the `list-managed-scheduled-executor-services` subcommand.

#### Example 14-11 Listing Managed Scheduled Executor Services

This example lists managed scheduled executor service resources on the default server instance, `server`.

```
asadmin> list-managed-scheduled-executor-services
concurrent/_defaultManagedScheduledExecutorService
concurrent/ScheduledExecutor1
concurrent/ScheduledExecutor2
Command list-managed-scheduled-executor-services executed successfully.
```

#### See Also

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

## To Update a Managed Scheduled Executor Service

You can change all of the settings for an existing managed scheduled executor service resource except its JNDI name. Use the `get` and `set` subcommands to view and change the values of the managed scheduled executor service attributes.



When a resource is updated, the existing resource is shut down and recreated. If applications used the resource prior to the update, the application or the server must be restarted.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the managed scheduled executor service resources by using the `list-managed-scheduled-executor-services` subcommand.
3. View the attributes of a managed scheduled executor service by using the `get` subcommand. For example:

```
asadmin> get resources.managed-scheduled-executor-
service.concurrent/ScheduledExecutor1.*
```

4. Set an attribute of the managed scheduled executor service by using the `set` subcommand. For example:

```
asadmin> set resources.managed-scheduled-executor-
```

```
service.concurrent/ScheduledExecutor1.deployment-order=120
```

## To Delete a Managed Scheduled Executor Service

Use the `delete-managed-scheduled-executor-service` subcommand in remote mode to delete an existing managed scheduled executor service. Deleting a managed scheduled executor service is a dynamic event and does not require server restart.

Before deleting a managed scheduled executor service resource, all associations to the resource must be removed.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the managed scheduled executor service resources by using the `list-managed-scheduled-executor-service` subcommand.
3. If necessary, notify users that the managed scheduled executor service is being deleted.
4. Delete the managed scheduled executor service by using the `delete-managed-scheduled-executor-service` subcommand.

### Example 14-12 Deleting a Managed Scheduled Executor Service

This example deletes the managed scheduled executor service resource named `concurrent/ScheduledExecutor1`.

```
asadmin> delete-managed-scheduled-executor-service concurrent/ScheduledExecutor1
Managed scheduled executor service concurrent/ScheduledExecutor1 deleted successfully.
Command delete-managed-scheduled-executor-service executed successfully.
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-managed-scheduled-executor-service` at the command line.

# 15 Administering the Object Request Broker (ORB)

Eclipse GlassFish supports a standard set of protocols and formats that ensure interoperability. Among these protocols are those defined by CORBA. The Object Request Broker (ORB) is the central component of CORBA. The ORB provides the required infrastructure to identify and locate objects, handle connection management, deliver data, and request communication. This chapter describes how to configure the ORB and the IIOP listeners.

The following topics are addressed here:

- [About the ORB](#)
- [Configuring the ORB](#)
- [Administering IIOP Listeners](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About the ORB

The Common Object Request Broker Architecture (CORBA) model is based on clients requesting services from distributed objects or servers through a well-defined interface by issuing requests to the objects in the form of remote method requests. A remote method request carries information about the operation that needs to be performed, including the object name (called an object reference) of the service provider and parameters, if any, for the invoked method. CORBA automatically handles network programming tasks such as object registration, object location, object activation, request de-multiplexing, error-handling, marshalling, and operation dispatching.

## Configuring the ORB

A CORBA object never talks directly with another. Instead, the object makes requests through a remote stub to the Internet Inter-Orb Protocol (IIOP) running on the local host. The local ORB then passes the request to an ORB on the other host using IIOP. The remote ORB then locates the appropriate object, processes the request, and returns the results.

IIOP can be used as a Remote Method Invocation (RMI) protocol by applications or objects using RMI-IIOP. Remote clients of enterprise beans (EJB modules) communicate with Eclipse GlassFish by using RMI-IIOP.

# Administering IIOP Listeners

An IIOP listener is a listen socket that accepts incoming connections from the remote clients of enterprise beans and from other CORBA-based clients. Multiple IIOP listeners can be configured for Eclipse GlassFish. For each listener, specify a port number (optional; default 1072), a network address, and security attributes (optional). If you create multiple listeners, you must assign a different port number for each listener.

The following topics are addressed here:

- [To Create an IIOP Listener](#)
- [To List IIOP Listeners](#)
- [To Update an IIOP Listener](#)
- [To Delete an IIOP Listener](#)

## To Create an IIOP Listener

Use the `create-iiop-listener` subcommand in remote mode to create an IIOP listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create an IIOP listener by using the `create-iiop-listener` subcommand.

Information about the properties for the subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 15-1 Creating an IIOP Listener

This example creates an IIOP listener named `sample_iiop_listener`.

```
asadmin> create-iiop-listener --listeneraddress 192.168.1.100  
--iiopport 1400 sample_iiop_listener  
Command create-iiop-listener executed successfully.
```

### See Also

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

## To List IIOP Listeners

Use the `list-iiop-listeners` subcommand in remote mode to list the existing IIOP listeners.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the IIOP listeners by using the `list-iiop-listeners` subcommand.

#### Example 15-2 Listing IIOP Listeners

This example lists all the IIOP listeners for the server instance.

```
asadmin> list-iiop-listeners
orb-listener-1
SSL
SSL_MUTUALAUTH
sample_iiop_listener
Command list-iiop-listeners executed successfully.
```

#### See Also

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

## To Update an IIOP Listener

1. List the IIOP listeners by using the `list-iiop-listeners` subcommand.
2. Modify the values for the specified IIOP listener by using the `set` subcommand.

The listener is identified by its dotted name.

#### Example 15-3 Updating an IIOP Listener

This example changes SSL from enabled to disabled.

```
asadmin> set "server.iiop-service.iiop-listener.SSL.enabled"
server.iiop-service.iiop-listener.SSL.enabled=false
Command set executed successfully.
```

## To Delete an IIOP Listener

Use the `delete-iiop-listener` subcommand in remote mode to delete an IIOP listener.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the IIOP listeners by using the `list-iiop-listeners` subcommand.
3. Delete an IIOP listener by using the `delete-iiop-listener` subcommand.
4. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 15-4 Deleting an IIOP Listener

This example deletes the IIOP listener named `sample_iiop_listener`.

```
asadmin> delete-iiop-listener sample_iiop_listener
Command delete-iiop-listener executed successfully.
```

#### See Also

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

# 16 Administering the JavaMail Service

Eclipse GlassFish includes the JavaMail API along with JavaMail service providers that allow an application component to send email notifications over the Internet and to read email from IMAP and POP3 mail servers.

The following topics are addressed here:

- [About JavaMail](#)
- [Administering JavaMail Resources](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About JavaMail

The JavaMail API is a set of abstract APIs that model a mail system. The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications and provide facilities for reading and sending electronic messages. Service providers implement particular protocols. Using the API you can add email capabilities to your applications. JavaMail provides access from Java applications to Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP) capable mail servers on your network or the Internet. The API does not provide mail server functionality; you must have access to a mail server to use JavaMail.

The JavaMail API is implemented as an optional package in the Java platform and is also available as part of the Jakarta EE platform.

To learn more about the JavaMail API, consult the [JavaMail web site](#).

## Administering JavaMail Resources

When you create a mail session, the server-side components and applications are enabled to access JavaMail services with JNDI, using the session properties you assign for them. When creating a mail session, you can designate the mail hosts, the transport and store protocols, and the default mail user so that components that use JavaMail do not have to set these properties. Applications that are heavy email users benefit because Eclipse GlassFish creates a single session object and makes the session available to any component that needs it.

JavaMail settings such as the following can be specified:

- JNDI Name. The unique name for the mail session. Use the naming sub-context prefix `mail/` for JavaMail resources. For example: `mail/MySession`
- Mail Host. The host name of the default mail server. The connect methods of the store and

transport objects use this value if a protocol-specific host property is not supplied. The name must be resolvable to an actual host name.

- Default User. The default user name to provide when connecting to a mail server. The connect methods of the store and transport objects use this value if a protocol-specific username property is not supplied.
- Default Return Address. The email address of the default user, in the form: `username@host.domain`.
- Description. A descriptive statement for the component.
- Session. Indicates whether or not mail session is enabled or disabled at this time

The following topics are addressed here:

- [To Create a JavaMail Resource](#)
- [To List JavaMail Resources](#)
- [To Update a JavaMail Resource](#)
- [To Delete a JavaMail Resource](#)

## To Create a JavaMail Resource

Use the `create-javamail-resource` subcommand in remote mode to create a JavaMail session resource. The JNDI name for a JavaMail session resource customarily includes the mail/ naming subcontext, For example: `mail/MyMailSession`.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a JavaMail resource by using the `create-javamail-resource` subcommand.

Information about the properties for the subcommand is included in this help page.

3. To apply your changes, restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 16-1 Creating a JavaMail Resource

This example creates a JavaMail resource named `mail/MyMailSession`. The escape character (\) is used in the `--fromaddress` option to distinguish the dot (.) and at sign (@).

```
asadmin> create-javamail-resource --mailhost localhost  
--mailuser sample --fromaddress sample\@sun\.com mail/MyMailSession  
Command create-javamail-resource executed successfully.
```

### See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help create-`

`javamail-resource` at the command line.

## To List JavaMail Resources

Use the `list-javamail-resources` subcommand in remote mode to list the existing JavaMail session resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the JavaMail resources by using the `list-javamail-resources` subcommand.

### Example 16-2 Listing JavaMail Resources

This example lists the JavaMail resources on `localhost`.

```
asadmin> list-javamail-resources  
mail/MyMailSession  
Command list-javamail-resources executed successfully.
```

### See Also

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

## To Update a JavaMail Resource

1. List the JavaMail resources by using the `list-javamail-resources` subcommand.
2. Modify the values for the specified JavaMail source by using the `set` subcommand.

The resource is identified by its dotted name.

### Example 16-3 Updating a JavaMail Resource

This example changes `joeserver` to `joe`.

```
asadmin> set server.resources.mail-resource.mail/  
MyMailSession.user=joeserver.resources.mail-resource.mail/  
MyMailSession.user=joe  
Command set executed successfully.
```

## To Delete a JavaMail Resource

Use the `delete-javamail-resource` subcommands in remote mode to delete a JavaMail session

resource.

## Before You Begin

References to the specified resource must be removed before running the `delete-javamail-resource` subcommands.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the JavaMail resources by using the `list-javamail-resources` subcommands.
3. Delete a JavaMail resource by using the `delete-javamail-resource` subcommands.
4. To apply your changes, restart Eclipse GlassFish.

[See To Restart a Domain.](#)

### Example 16-4 Deleting a JavaMail Resource

This example deletes the JavaMail session resource named `mail/MyMailSession`.

```
asadmin> delete-javamail-resource mail/MyMailSession
Command delete-javamail-resource executed successfully.
```

## See Also

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

# 17 Administering the Java Message Service (JMS)

The Java Message Service (JMS) API is a messaging standard that allows Jakarta EE applications and components, including message-driven beans (MDBs), to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

Eclipse GlassFish supports JMS messaging by communicating with a JMS provider through a Jakarta EE Connector resource adapter. By default, Eclipse GlassFish provides JMS messaging through its built-in jmsra resource adapter communicating with Open Message Queue, which is included with Eclipse GlassFish. This combination, known as the JMS Service, is tightly integrated with Eclipse GlassFish, providing a rich set of `asadmin` subcommands and Administration Console pages to simplify JMS messaging administration tasks.

Eclipse GlassFish also supports the Generic Resource Adapter for JMS (GenericJMSRA) for use as a resource adapter to connect to other JMS providers. The last section in this chapter, [Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers](#), describes the GenericJMSRA and provides instructions for using it to make other supported JMS providers available to Eclipse GlassFish.

The following topics are addressed here:

- [About the JMS Service](#)
- [Updating the JMS Service Configuration](#)
- [Administering JMS Hosts](#)
- [Administering JMS Connection Factories and Destinations](#)
- [Administering JMS Physical Destinations](#)
- [Special Situations When Using the JMS Service](#)
- [Troubleshooting the JMS Service](#)
- [Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers](#)

Instructions for accomplishing the task in this chapter by using the Administration Console are contained in the Administration Console online help.

## About the JMS Service

To support JMS messaging, the JMS Service provides the following administrative objects:

### JMS Service Configuration

The JMS service configuration is part of the overall configuration for a GlassFish standalone instance or cluster. It specifies how the JMS Service is to create and maintain connections with JMS Hosts.

## JMS Hosts

JMS hosts are the message servers that host destinations, store messages, and interact with applications to send and receive messages across connections. In Message Queue, JMS hosts are called brokers.

The JMS service supports these types of JMS hosts:

- Embedded type, in which the JMS host runs in the same JVM as the GlassFish instance; its configuration and lifecycle are managed by the JMS service.
- Local type, in which the JMS host runs separately on the same host as the GlassFish instance; its configuration and lifecycle are managed by the JMS service.
- Remote type, in which the JMS host represents a Message Queue broker or broker cluster that is external to the JMS service; its operation is managed using Message Queue administrative tools.

For more information about JMS host types, see [About JMS Host Types](#).

## JMS Connection Factory Resources

JMS connection factory resources house the information that applications use to connect to a JMS provider. For each JMS connection factory, the JMS service automatically maintains a GlassFish connector resource and a GlassFish connector connection pool in order to support connection pooling and failover.

## JMS Destination Resources

JMS destination resources house the information that applications use to specify the target destination of messages they produce and the source destination of messages they consume. For each JMS destination resource, the JMS service automatically maintains a GlassFish administered object.

## JMS Physical Destinations

JMS physical destinations provide a means to create and manage JMS destinations administratively instead of having them created dynamically when needed by an application. While dynamic creation of destinations is often sufficient during application development, administratively created physical destinations are more suitable for production environments.

## JMS Service High Availability

Just as Eclipse GlassFish supports clusters of instances to provide high availability, Message Queue supports clusters of brokers to provide service availability or service and data availability, depending on the type of broker cluster, as described in "[Broker Clusters](#)" in Open Message Queue Technical Overview.

The JMS service takes advantage of this Message Queue capability and automatically creates and manages a Message Queue broker cluster when a GlassFish cluster's configuration specifies Embedded or Local type JMS hosts. Additionally, both GlassFish clusters and standalone instances can use Message Queue broker clusters as Remote type JMS hosts.

For information about how the JMS service supports GlassFish clusters and Message Queue broker clusters, see "[Configuring Java Message Service High Availability](#)" in Eclipse GlassFish High Availability Administration Guide.

## Updating the JMS Service Configuration

Because the JMS service configuration is part of the overall configuration for a standalone instance or cluster, it is created when the standalone instance or cluster is created. You can then update the JMS service configuration by using the Java Message Service page for the configuration in the Administration Console, or by using a `set` subcommand of the following form:

```
set configs.config.config-name.jms-service.attribute-name=attribute-value
```

The attributes you can set are:

### **type**

The JMS host type the service is to use. Available choices are `EMBEDDED`, `LOCAL` and `REMOTE`. See [About JMS Host Types](#) for more information.

### **init-timeout-in-seconds**

The number of seconds Eclipse GlassFish waits for the JMS service to start before aborting the startup.

### **start-args**

A list of arguments the JMS service passes to Embedded and Local type JMS hosts on startup. Permissible arguments are the options supported by the Message Queue `imqbrokerd` command, as described in "[Broker Utility](#)" in Open Message Queue Administration Guide.

### **default-jms-host**

The name of the default JMS host.

### **reconnect-enabled**

When set to `true`, the JMS service attempts to reconnect to a JMS host (or one of the JMS hosts in the AddressList) when a connection is lost.

### **reconnect-attempts**

The number of attempts to connect (or reconnect) for each JMS host in the AddressList before the JMS service tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the JMS service attempts to connect to the first address until it succeeds).

### **reconnect-interval-in-seconds**

The number of seconds between reconnect attempts. This interval applies for attempts on each JMS host in the AddressList and for successive addresses in the list. If it is too short, this time interval does not give a JMS host time to recover. If it is too long, the reconnect might represent

an unacceptable delay.

#### addresslist-behavior

The order of connection attempts. Available choices are:

##### random

Select a JMS host from the AddressList randomly. If there are many clients attempting a connection using the same connection factory, specify `random` to prevent them from all being connected to the same JMS host.

##### priority

Always try to connect to the first JMS host in the AddressList and use another one only if the first one is not available.

#### addresslist-iterations

The number of times the JMS service iterates through the AddressList in an effort to establish (or reestablish) a connection. A value of -1 indicates that the number of attempts is unlimited.

#### mq-scheme

#### mq-service

The Message Queue address scheme name and connection service name to use for the AddressList if a non-default scheme or service is to be used. See "[Connection Handling](#)" in Open Message Queue Administration Guide for syntax information.



After making changes to the JMS service configuration, Eclipse GlassFish instances that use the configuration must be restarted in order for the changes to be propagated.

## Setting Message Queue Broker Properties in the JMS Service Configuration

You can specify any Message Queue broker property in the JMS service configuration by adding it by name to the Additional Properties table on the Java Message Service page for the configuration in the Administration Console, or by using a `set` subcommand of the following form:

```
set configs.config.config-name.jms-service.property.broker-property-name=value
```

If the broker property name includes dots, preface the dots with two backslashes (\\\); for example, to set the `imq.system.max_count` property, specify `imq\\.system\\.max_count` in the `set` subcommand.



You can also set broker properties in the JMS host. If you set the same broker property in both the JMS service configuration and the JMS host, the value specified in the JMS host is used.

# Administering JMS Hosts

A JMS host represents a Message Queue broker. JMS contains a JMS hosts list (the [AddressList](#) property) that contains all the JMS hosts that are used by Eclipse GlassFish. The JMS hosts list is populated with the hosts and ports of the specified Message Queue brokers and is updated whenever a JMS host configuration changes. When you create JMS resources or deploy message driven beans, the resources or beans inherit the JMS hosts list.

The following topics are addressed here:

- [About JMS Host Types](#)
- [Configuring Embedded and Local JMS Hosts](#)
- [To Create a JMS Host](#)
- [To List JMS Hosts](#)
- [To Update a JMS Host](#)
- [To Delete a JMS Host](#)

For information about administering JMS hosts that are servicing GlassFish clusters, see ["Configuring GlassFish Clusters to Use Message Queue Broker Clusters"](#) in Eclipse GlassFish High Availability Administration Guide.

## About JMS Host Types

The JMS service uses Message Queue (MQ) brokers as JMS hosts, integrating them in three ways:

### Embedded Type

When the JMS service configuration's `type` attribute is `EMBEDDED`, the MQ broker is co-located in the same JVM as the Eclipse GlassFish instance it services. The JMS service starts it in-process and manages its configuration and lifecycle.

For this type, the JMS service uses lazy initialization to start the broker when the first JMS operation is requested instead of immediately when the GlassFish instance is started. If necessary, you can force startup of the broker by using the `jms-ping` command.

Additionally, if the GlassFish instance is a standalone instance (not a clustered instance), JMS operations use a Message Queue feature called direct mode to bypass the networking stack, leading to performance optimization.

### Local Type

When the JMS service configuration's `type` attribute is `LOCAL`, the JMS service starts the MQ broker specified in the configuration as the default JMS host in a separate process on the same host as the Eclipse GlassFish instance. The JMS service manages its configuration and lifecycle.

For this type, the JMS service starts the broker immediately when the GlassFish instance is started.

The JMS service provides the Message Queue broker an additional port to start the RMI registry. This port number is equal to the broker's JMS port plus 100. For example, if the JMS port number is 37676, then the additional port's number will be 37776. Additionally, the `start-args` property

of the JMS service configuration can be used to specify Message Queue broker startup options.

## Remote Type

When the JMS service configuration's `type` attribute is `REMOTE`, the JMS service uses the information defined by the default JMS host to communicate with an MQ broker or broker cluster that has been configured and started using Message Queue tools, as described in the [Open Message Queue Administration Guide](#). Ongoing administration and tuning of the broker or broker cluster are also performed using Message Queue tools.

## Configuring Embedded and Local JMS Hosts

Because the JMS service, not Message Queue, manages Embedded and Local JMS hosts automatically, you should avoid using Message Queue utilities to configure them. Instead, specify broker properties in the JMS service configuration or in the JMS host.

Should the need to use Message Queue utilities arise, you must use the `-varhome` option when running certain Message Queue utilities to specify the `IMQ_VARHOME` location of the Embedded or Local JMS host. This location depends on which GlassFish instance the JMS host is servicing:

- For `server`, the Domain Administration Server (DAS), the `IMQ_VARHOME` location is:

```
domain-root-dir/domain-dir/imq
```

- For any other GlassFish instance, the `IMQ_VARHOME` location is:

```
as-install/nodes/node-name/instance-name/imq
```

For example, the broker log file for an Embedded or Local JMS host servicing the DAS is available at `domain-root-dir/domain-dir/imq/instances/imqbroker/log/log.txt`, and the broker log file for an Embedded or Local JMS host servicing any other GlassFish instance is available at `as-install/nodes/node-name/instance-name`/imq/instances/mq-instance-name/log/log.txt```.

 When using Message Queue utilities on the Windows platform, you must explicitly use the Windows executable (`.exe`) versions of the utilities, even when running command shells such as Cygwin. For example, instead of running `imqcmd`, you must run `imqcmd.exe`.

## To Create a JMS Host

The default JMS service configuration includes a JMS host, `default_JMS_host`. For most situations, this host is sufficient, so replacing it or creating additional JMS hosts is not often necessary and is a task for advanced users. Use the `create-jms-host` subcommand in remote `asadmin` mode to create an additional JMS host.

## 1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

## 2. Create the JMS host by using the `create-jms-host` subcommand:

```
asadmin> create-jms-host --mqhost hostName --mqport portNumber  
--mquser adminUser --mqpassword adminPassword --target glassfishTarget  
--property mqBrokerPropList --force trueFalse jms-host-name
```

### **--mqhost**

The host name of the Message Queue broker.

### **--mqport**

The port number of the Message Queue broker.

### **--mquser**

The user name of the administrative user of the Message Queue broker.

### **--mqpassword**

The password of the administrative user of the Message Queue broker.

### **--target**

The Eclipse GlassFish object for which the JMS host is being created. For details, see [create-jms-host\(1\)](#).

### **--property**

A list of one or more Message Queue broker properties to configure the broker. The list is colon-separated (`:`) and has the form:

```
prop1Name=prop1Value:prop2Name=prop2Value:...
```

If a broker property name includes dots, preface the dots with two backslashes (`\\"`); for example, to include the `imq.system.max_count` property, specify `imq\\.\system\\.\max_count` in the `--property` option.



You can also set broker properties in the JMS service configuration. If you set the same broker property in both the JMS host and the JMS service configuration, the value specified in the JMS host is used.

### **--force**

Specifies whether the subcommand overwrites the existing JMS host of the same name. The default value is `false`.

### **jms-host-name**

The unique name of the JMS host.

## Example 17-1 Creating a JMS Host

This example creates a JMS host named `MyNewHost`.

```
asadmin> create-jms-host --mqhost pigeon --mqport 7677  
--mquser admin --mqpassword admin MyNewHost  
Jms Host MyNewHost created.  
Command create-jms-host executed successfully.
```

### See Also

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

## To List JMS Hosts

Use the `list-jms-hosts` subcommand in remote `asadmin` mode to list the existing JMS hosts.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the JMS hosts by using the `list-jms-hosts` subcommand.

### Example 17-2 Listing JMS Hosts

The following subcommand lists the existing JMS hosts.

```
asadmin> list-jms-hosts  
default_JMS_host  
MyNewHost  
Command list-jmsdest executed successfully
```

## To Update a JMS Host

Use the `set` subcommand in remote `asadmin` mode to update an existing JMS host.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. Use the `get` subcommand to list the current attribute values of the desired JMS host:

```
asadmin> get configs.config.config-name.jms-service.jms-host.jms-host-name.*
```

For information about JMS host attributes, see [create-jms-host\(1\)](#).

3. Use the `set` subcommand to modify a JMS host attribute:

```
asadmin> set configs.config.config-name.jms-service.jmshost.  
jms-host-name.attribute-name=attribute-value
```

### The attributes you can set are

#### `host`

The host name of the Message Queue broker.

#### `port`

The port number of the Message Queue broker.

#### `admin-user-name`

The user name of the administrative user of the Message Queue broker.

#### `admin-password`

The password of the administrative user of the Message Queue broker.

#### ``property.`broker-property-name`

A Message Queue broker property. The property, and the value assigned to it, are used to configure the Message Queue broker.

If the broker property name includes dots, preface the dots with two backslashes ('\\'); for example, to include the 'imq.system.max\_count' property, specify 'imq\\.system\\.max\_count' in the 'set' subcommand.



You can also set broker properties in the JMS service configuration. If you set the same broker property in both the JMS host and the JMS service configuration, the value specified in the JMS host is used.

### Example 17-3 Updating a JMS Host

This example changes the value of the `host` attribute of the JMS host `default_JMS_Host`. By default this value is `localhost`.

```
asadmin> set configs.config.server-config.jms-service.jms-host.default_JMS_host.host=  
"server1.middleware.example.com"
```

### To Delete a JMS Host

Use the `delete-jms-host` subcommand in remote `asadmin` mode to delete a JMS host from the JMS

service. If you delete the only JMS host, the JMS service will not be able to start until you create a new JMS host.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the JMS hosts by using the `list-jms-hosts` subcommand.
3. Delete a JMS host by using the `delete-jms-host` subcommand.

#### Example 17-4 Deleting a JMS Host

This example deletes a JMS host named `MyNewHost`.

```
asadmin> delete-jms-host MyNewHost
Command delete-jms-host executed successfully.
```

#### See Also

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

## Administering JMS Connection Factories and Destinations

The JMS API uses two kinds of administered objects. Connection factory objects allow an application to create other JMS objects programmatically. Destination objects serve as repositories for messages. How these objects are created is specific to each implementation of JMS. In Eclipse GlassFish, JMS is implemented by performing the following tasks:

- Creating a connection factory
- Creating a destination, which requires creating a physical destination and a destination resource that refers to the physical destination

JMS applications use the Java Naming and Directory Interface (JNDI) API to access the connection factory and destination resources. A JMS application normally uses at least one connection factory and at least one destination. By studying the application or consulting with the application developer, you can determine what resources must be created. The order in which the resources are created does not matter.

The Jakarta EE standard specifies that certain default resources be made available to applications, and defines specific JNDI names for these default resources. Eclipse GlassFish makes these names available through the use of logical JNDI names, which map Jakarta EE standard JNDI names to specific Eclipse GlassFish resources. For JMS connection factory resources, the Jakarta EE standard name `java:comp/DefaultJMSConnectionFactory` is mapped to the `jms/_defaultConnectionFactory` resource.

Eclipse GlassFish provides the following types of connection factory objects:

- `QueueConnectionFactory` objects, used for point-to-point communication
- `TopicConnectionFactory` objects, used for publish-subscribe communication
- `ConnectionFactory` objects, which can be used for both point-to-point and publish-subscribe communications (recommended for new applications)

Eclipse GlassFish provides the following types of destination objects:

- `Queue` objects, used for point-to-point communication
- `Topic` objects, used for publish-subscribe communication

The following topics are addressed here:

- [To Create a Connection Factory or Destination Resource](#)
- [To List JMS Resources](#)
- [To Delete a Connection Factory or Destination Resource](#)

The subcommands in this section can be used to administer both the connection factory resources and the destination resources. For information on JMS service support of connection pooling and failover, see "[Connection Failover](#)" in Eclipse GlassFish High Availability Administration Guide. For instructions on administering physical destinations, see [Administering JMS Physical Destinations](#).

## To Create a Connection Factory or Destination Resource

For each JMS connection factory that you create, Eclipse GlassFish creates a connector connection pool and connector resource. For each JMS destination that you create, Eclipse GlassFish creates a connector admin object resource. If you delete a JMS resource, Eclipse GlassFish automatically deletes the connector resources.

Use the `create-jms-resource` command in remote `asadmin` mode to create a JMS connection factory resource or a destination resource.



To specify the `addresslist` property (in the format `host:mqport,host2:mqport,host3:mqport`) for the `asadmin create-jms-resource` command, escape the `:` by using `\`. For example, `host1\\:mqport,host2\\:mqport,host3\\:mqport`. For more information about using escape characters, see the [oasadmin\(1M\)](#) help page.

To update a JMS connection factory, use the `set` subcommand for the underlying connector connection pool. See [To Update a Connector Connection Pool](#).

To update a destination, use the `set` subcommand for the admin object resource. See [To Update an Administered Object](#).

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. Create a JMS resource by using the `create-jms-resource` command.

Information about the properties for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

#### Example 17-5 Creating a JMS Connection Factory

This example creates a connection factory resource of type `jakarta.jms.ConnectionFactory` whose JNDI name is `jms/DurableConnectionFactory`. The `ClientId` property sets a client ID on the connection factory so that it can be used for durable subscriptions. The JNDI name for a JMS resource customarily includes the `jms/` naming subcontext.

```
asadmin> create-jms-resource --restype jakarta.jms.ConnectionFactory  
--description "connection factory for durable subscriptions"  
--property ClientId=MyID jms/DurableConnectionFactory  
Command create-jms-resource executed successfully.
```

#### Example 17-6 Creating a JMS Destination

This example creates a destination resource whose JNDI name is `jms/MyQueue`.

```
asadmin> create-jms-resource --restype jakarta.jms.Queue  
--property Name=PhysicalQueue jms/MyQueue  
Command create-jms-resource executed successfully.
```

#### See Also

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

## To List JMS Resources

Use the `list-jms-resources` subcommand in remote `asadmin` mode to list the existing connection factory and destination resources.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the existing JMS resources by using the `list-jms-resources` subcommand.

## Example 17-7 Listing All JMS Resources

This example lists all the existing JMS connection factory and destination resources.

```
asadmin> list-jms-resources
jms/Queue
jms/ConnectionFactory
jms/DurableConnectionFactory
jms/Topic
Command list-jms-resources executed successfully
```

## Example 17-8 Listing a JMS Resources of a Specific Type

This example lists the resources for the resource type `javax`.

```
asadmin> list-jms-resources --restype javax.jms.TopicConnectionFactory
jms/DurableTopicConnectionFactory
jms/TopicConnectionFactory
Command list-jms-resources executed successfully.
```

## See Also

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

## To Delete a Connection Factory or Destination Resource

Use the `delete-jms-resource` subcommand in remote `asadmin` mode to remove the specified connection factory or destination resource.

### Before You Begin

Ensure that you remove all references to the specified JMS resource before running this subcommand.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the existing JMS resources by using the `list-jms-resources` subcommand.
3. Delete the JMS resource by using the `delete-jms-resource` subcommand.

## Example 17-9 Deleting a JMS Resource

This example deletes the `jms/Queue` resource.

```
asadmin> delete-jms-resource jms/Queue  
Command delete-jms-resource executed successfully
```

## See Also

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

# Administering JMS Physical Destinations

Messages are delivered for routing and delivery to consumers by using physical destinations in the JMS provider. A physical destination is identified and encapsulated by an administered object (such as a [Topic](#) or [Queue](#) destination resource) that an application component uses to specify the destination of messages it is producing and the source of messages it is consuming. For instructions on configuring a destination resource, see [To Create a Connection Factory or Destination Resource](#).

If a message-driven bean is deployed and the physical destination it listens to does not exist, Eclipse GlassFish automatically creates the physical destination and sets the value of the `maxNumActiveConsumers` property to `-1`. However, it is good practice to create the physical destination beforehand. The first time that an application accesses a destination resource, Message Queue automatically creates the physical destination specified by the Name property of the destination resource. This automatically created physical destination is temporary and expires after a period specified by a Message Queue configuration property, provided that there are no messages in it and no message producers or consumers connected to it.

The following topics are addressed here:

- [To Create a JMS Physical Destination](#)
- [To List JMS Physical Destinations](#)
- [To Purge Messages From a Physical Destination](#)
- [To Delete a JMS Physical Destination](#)

## To Create a JMS Physical Destination

For production purposes, always create physical destinations. During the development and testing phase, however, this step is not required. Use the `create-jmsdest` subcommand in remote `asadmin` mode to create a physical destination.

Because a physical destination is actually a Message Queue object rather than a server object, you use Message Queue broker commands to update properties. For information on Message Queue properties, see the [Open Message Queue Administration Guide](#).

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. Create a JMS physical destination by using the `create-jmsdest` subcommand.

Information about the properties for the subcommand is included in this help page.

3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

#### Example 17-10 Creating a JMS Physical Destination

This example creates a queue named `PhysicalQueue`.

```
asadmin> create-jmsdest --desttype queue --property  
User=public:Password=public PhysicalQueue  
Command create-jmsdest executed successfully.
```

#### See Also

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

## To List JMS Physical Destinations

Use the `list-jmsdest` subcommand in remote `asadmin` mode to list the existing JMS physical destinations.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the existing JMS physical destinations by using the `list-jmsdest` subcommand.

#### Example 17-11 Listing JMS Physical Destinations

This example lists the physical destinations for the default server instance.

```
asadmin> list-jmsdest  
PhysicalQueue queue {}  
PhysicalTopic topic {}  
Command list-jmsdest executed successfully.
```

#### See Also

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

## To Purge Messages From a Physical Destination

Use the `flush-jmsdest` subcommand in remote `asadmin` mode to purge the messages from a physical destination in the specified target's JMS service configuration.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. Purge messages from the a JMS physical destination by using the `flush-jmsdest` subcommand.
3. If needed, restart the server.

Some properties require server restart. See [Configuration Changes That Require Restart](#). If your server needs to be restarted, see [To Restart a Domain](#).

### Example 17-12 Flushing Messages From a JMS Physical Destination

This example purges messages from the queue named `PhysicalQueue`.

```
asadmin> flush-jmsdest --desttype queue PhysicalQueue
Command flush-jmsdest executed successfully
```

### See Also

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

## To Delete a JMS Physical Destination

Use the `delete-jmsdest` subcommand in remote `asadmin` mode to remove the specified JMS physical destination.

1. Ensure that the server is running.

Remote `asadmin` subcommands require a running server.

2. List the existing JMS physical destinations by using the `list-jmsdest` subcommand.
3. Delete the physical resource by using the `delete-jmsdest` subcommand.

### Example 17-13 Deleting a Physical Destination

This example deletes the queue named `PhysicalQueue`.

```
asadmin> delete-jmsdest --desttype queue PhysicalQueue
```

Command `delete-jmsdest` executed successfully

## See Also

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

# Special Situations When Using the JMS Service

As mentioned earlier, Message Queue, through the built-in `jmsra` resource adapter, is tightly integrated with Eclipse GlassFish to provide JMS messaging managed through a rich set of `asadmin` subcommands and Administration Console pages to simplify JMS messaging administration tasks. In most instances, this tight integration is transparent and automatic, requiring no special effort on the part of an administrator. In certain special situations, though, an administrator must perform a task such as setting a Message Queue broker property or a GlassFish object attribute to enable or disable a capability of the integration. The topics in this section describe these situations.

## Restarting an Embedded or Local Broker That Has Failed

Because the JMS service, not Message Queue, manages the lifecycle of brokers acting as Embedded and Local JMS hosts, do not use the `imqbrokerd` Message Queue utility to start such a broker that has failed. Instead, restart the GlassFish instance that the broker is servicing.

## Changing the Admin User Password for an Embedded or Local Broker

Follow these steps to change the `admin` user password for an Embedded or Local broker:

1. Make sure the broker is running.
2. Use the `imqusermgr` Message Queue utility to change the password of the `admin` user.
3. Edit the configuration of the JMS host, changing the password of the `admin` user to the new password.
4. Restart the GlassFish instance that the broker is servicing.

When changing the password for the brokers in a broker cluster, first perform steps 1 and 2 on each broker. Then, perform step 3. Finally, perform step 4 on each broker.

Using SSL to Connect to an Oracle Internet Directory (OID) or Oracle Virtual Directory (OVD) User Repository:: When using SSL to connect to an OID or OVD user repository, you must set the `imq.user_repository.ldap.ssl.socketfactory` Message Queue broker property to `com.sun.enterprise.security.auth.realm.ldap.CustomSocketFactory`.

# Troubleshooting the JMS Service

If you encounter problems, consider the following:

- Use the `jms-ping` subcommand to confirm that the Message Queue broker is running.

- View the Eclipse GlassFish log file. For `server`, the Domain Administrations Server (DAS), the log is available at `domain-dir/logs/server.log`; for other GlassFish instances, the log is available at `as-install/nodes/node-name/instance-name/logs/server.log`.

If the log file indicates that a Message Queue broker acting as a Remote JMS host did not respond to a message, stop the broker and then restart it.

- View the broker log. For a broker associated with the Domain Administration Server (DAS), the log is available at `domain-dir`/imq/instances/imqbroker/log/log.txt``; for brokers associated with other GlassFish instances, the log is available at `as-install/nodes/node-name/instance-name`/imq/instances/mq-instance-name/log/log.txt```.
- For Remote type JMS hosts, be sure to start Message Queue brokers first, then Eclipse GlassFish instances.
- If all Message Queue brokers are down, it can take up to 30 minutes for Eclipse GlassFish to go down or up when you are using the default values in JMS. You can change the default values for this timeout. For example:

```
asadmin set domain1.jms-service.reconnect-interval-in-seconds=5
```

## Using the Generic Resource Adapter for JMS to Integrate Supported External JMS Providers

Eclipse GlassFish supports the integration and use of Oracle WebLogic JMS and IBM WebSphere MQ JMS providers through the use of the Generic Resource Adapter for JMS (GenericJMSRA). This Jakarta EE connector 1.5 resource adapter can wrap the JMS client library of Oracle WebLogic JMS and IBM WebSphere MQ and make it available for use by GlassFish. The adapter is a `.rar` archive that can be deployed and configured using Eclipse GlassFish administration tools.

The following topics are addressed here:

- [Configuring GenericJMSRA for Supported External JMS Providers](#)
- [Using GenericJMSRA with WebLogic JMS](#)
- [Using GenericJMSRA with IBM WebSphere MQ](#)

## Configuring GenericJMSRA for Supported External JMS Providers

GenericJMSRA has three main properties that need to be configured: `SupportXA`, `DeliveryType`, and `ProviderIntegrationMode`. The values that need to be set for them depends on the capabilities of the JMS provider being used, as follows:

- `SupportXA` — indicates whether the JMS provider supports XA or not.
- `DeliveryType` — indicates whether an MDB should use a `ConnectionConsumer` or

`Consumer.receive()` when consuming messages.

- `ProviderIntegrationMode` — indicates what mode of integration is required. The available integration modes are jndi and javabean. Some JMS providers support only one integration mode while others may offer the choice of both
  - If jndi is specified, then the resource adapter will obtain JMS connection factories and destinations from the JMS provider's JNDI repository.
  - If javabean is specified then the resource adapter will obtain JMS connection factories and destinations by instantiating the appropriate classes directly.

Which option is specified determines which other properties need to be set.

## To Deploy and Configure GenericJMSRA

Before deploying GenericJMSRA, JMS client libraries must be made available to Eclipse GlassFish. For some JMS providers, client libraries might also include native libraries. In such cases, these native libraries must be made available to any Eclipse GlassFish JVMs.

1. Download the [genericra.rar](#) archive.
2. Deploy GenericJMSRA the same way you would deploy a connector module.

See "Deploying a Connector Module" in Eclipse GlassFish Application Deployment Guide.

3. Configure the resource adapter's properties.

See [GenericJMSRA Configuration Properties](#).

4. Create a connector connection pool.

See [To Create a Connector Connection Pool](#).

5. Create a connector resource.

See [To Create a Connector Resource](#).

6. Create an administered object resource.

See [To Create an Administered Object](#).

## GenericJMSRA Configuration Properties

The following table describes the properties that can be set to when configuring the resource adapter.

<b>Property Name</b>	<b>Valid Values</b>	<b>Default Value</b>	<b>Description</b>
<code>SupportsXA</code>	<code>true/false</code>	false	Specifies whether the JMS client supports XA transactions.
<code>DeliveryType</code>	<code>Synchronous/ Asynchronous</code>	Asynchronous	Specifies whether an MDB should use a <code>ConnectionConsumer</code> (Asynchronous) or <code>consumer.receive()</code> (Synchronous) when consuming messages.
<code>QueueConnectionFactoryClassName</code>	A valid class name	None	Class name of <code>jakarta.jms.QueueConnectionFactory</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
<code>TopicConnectionFactoryClassName</code>	A valid class name	None	Class name of <code>jakarta.jms.TopicConnectionFactory</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is specified as <code>javabean</code> .
<code>XAConnectionFactoryClassName</code>	A valid class name	None	Class name of <code>jakarta.jms.ConnectionFactory</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is specified as <code>javabean</code> .
<code>XAQueueConnectionFactoryClassName</code>	A valid class name	None	Class name of <code>jakarta.jms.XAQueueConnectionFactory</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is specified as <code>javabean</code> .
<code>XATopicConnectionFactoryClassName</code>	A valid class name	None	Class name of <code>jakarta.jms.XATopicConnectionFactory</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .

Property Name	Valid Values	Default Value	Description
<code>TopicClassName</code>	A valid class name	None	Class Name of <code>jakarta.jms.Topic</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
<code>QueueClassName</code>	A valid class name	None	Class Name of <code>jakarta.jms.Queue</code> implementation of the JMS client. This class must be made available on the application server classpath. Used if <code>ProviderIntegrationMode</code> is specified as a <code>javabean</code> .
<code>ConnectionFactoryProperties</code>	Name value pairs separated by comma	None	Specifies the <code>javabean</code> property names and values of the <code>ConnectionFactory</code> of the JMS client. Required only if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
<code>JndiProperties</code>	Name value pairs separated by comma	None	Specifies the JNDI provider properties to be used for connecting to the JMS provider's JNDI. Used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
<code>CommonSetterMethodName</code>	Method name	None	Specifies the common setter method name that some JMS vendors use to set the properties on their administered objects. Used only if <code>ProviderIntegrationMode</code> is <code>javabean</code> . For example, in the case of Message Queue, this would be <code>setProperty</code> .
<code>UserName</code>	Name of the JMS user	None	User name to connect to the JMS Provider.
<code>Password</code>	Password for the JMS user	None	Password to connect to the JMS provider.

Property Name	Valid Values	Default Value	Description
RMPolicy	ProviderManaged or OnePerPhysicalConnection	ProviderManaged	<p>The <code>isSameRM</code> method on an <code>XAResource</code> is used by the Transaction Manager to determine if the Resource Manager instance represented by two <code>XAResources</code> are the same. When <code>RMPolicy</code> is set to <code>ProviderManaged</code> (the default value), the JMS provider is responsible for determining the <code>RMPolicy</code> and the <code>XAResource</code> wrappers in GenericJMSRA merely delegate the <code>isSameRM</code> call to the JMS provider's XA resource implementations. This should ideally work for most JMS providers.</p> <p>Some <code>XAResource</code> implementations such as WebSphere MQ rely on a resource manager per physical connection and this causes issues when there is inbound and outbound communication to the same queue manager in a single transaction (for example, when an MDB sends a response to a destination). When <code>RMPolicy</code> is set to <code>OnePerPhysicalConnection</code>, the <code>XAResource</code> wrapper implementation's <code>isSameRM</code> in GenericJMSRA would check if both the <code>XAResources</code> use the same physical connection, before delegating to the wrapped objects.</p>

## Connection Factory Properties

`ManagedConnectionFactory` properties are specified when a `connector-connection-pool` is created. All the properties specified while creating the resource adapter can be overridden in a `ManagedConnectionFactory`. Additional properties available only in `ManagedConnectionFactory` are given below.

<b>Property Name</b>	<b>Valid Value</b>	<b>Default Value</b>	<b>Description</b>
<code>ClientId</code>	A valid client ID	None	<code>ClientID</code> as specified by JMS 1.1 specification.
<code>ConnectionFactoryJndiName</code>	JNDI Name	None	JNDI name of the connection factory bound in the JNDI tree of the JMS provider. The administrator should provide all connection factory properties (except <code>clientID</code> ) in the JMS provider itself. This property name will be used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
<code>ConnectionValidationEnabled</code>	true/false	false	If set to true, the resource adapter will use an exception listener to catch any connection exception and will send a <code>CONNECTION_ERROR_OCCURRED</code> event to application server.

## Destination Properties

Properties in this section are specified when a destination (queue or topic) is created. All the resource adapter properties can be overridden in a destination. Additional properties available only in the destination are given below.

<b>Property Name</b>	<b>Valid Value</b>	<b>Default Value</b>	<b>Description</b>
<code>DestinationJndiName</code>	JNDI Name	None	JNDI name of the destination bound in the JNDI tree of the JMS provider. The Administrator should provide all properties in the JMS provider itself. This property name will be used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
<code>DestinationProperties</code>	Name value pairs separated by a comma	None	Specifies the <code>javabean</code> property names and values of the destination of the JMS client. Required only if <code>ProviderIntegrationMode</code> is <code>javabean</code> .

## Activation Spec Properties

Properties in this section are specified in the Eclipse GlassFish `glassfish-ejb-jar.xml` deployment descriptor of an MDB as `activation-config-properties`. All the resource adapter properties can be overridden in an Activation Spec. Additional properties available only in ActivationSpec are given below.

<b>Property Name</b>	<b>Valid Value</b>	<b>Default Value</b>	<b>Description</b>
<code>MaxPoolSize</code>	An integer	8	Maximum size of server session pool internally created by the resource adapter for achieving concurrent message delivery. This should be equal to the maximum pool size of MDB objects.
<code>MaxWaitTime</code>	An integer	3	The resource adapter will wait for the time in seconds specified by this property to obtain a server session from its internal pool. If this limit is exceeded, message delivery will fail.
<code>SubscriptionDurability</code>	Durable or Non-Durable	Non-Durable	<code>SubscriptionDurability</code> as specified by JMS 1.1 specification.
<code>SubscriptionName</code>	+	None	<code>SubscriptionName</code> as specified by JMS 1.1 specification.

Property Name	Valid Value	Default Value	Description
MessageSelector	A valid message selector	None	MessageSelector as specified by JMS 1.1 specification.
ClientID	A valid client ID	None	ClientID as specified by JMS 1.1 specification.
ConnectionFactoryJndiName	A valid JNDI Name	None	JNDI name of connection factory created in JMS provider. This connection factory will be used by resource adapter to create a connection to receive messages. Used only if ProviderIntegrationMode is configured as <code>jndi</code> .
DestinationJndiName	A valid JNDI Name	None	JNDI name of destination created in JMS provider. This destination will be used by resource adapter to create a connection to receive messages from. Used only if ProviderIntegrationMode is configured as <code>jndi</code> .
DestinationType	<code>jakarta.jms.Queue</code> or <code>jakarta.jms.Topic</code>	Null	Type of the destination the MDB will listen to.
DestinationProperties	Name-value pairs separated by comma	None	Specifies the javabean property names and values of the destination of the JMS client. Required only if ProviderIntegrationMode is <code>javabean</code> .
RedeliveryAttempts	integer	+	Number of times a message will be delivered if a message causes a runtime exception in the MDB.
Redelivery``Interval	time in seconds	+	Interval between repeated deliveries, if a message causes a runtime exception in the MDB.

<b>Property Name</b>	<b>Valid Value</b>	<b>Default Value</b>	<b>Description</b>
<code>SendBadMessagesToDMD</code>	true/false	False	Indicates whether the resource adapter should send the messages to a dead message destination, if the number of delivery attempts is exceeded.
<code>DeadMessageDestinationJndiName</code>	a valid JNDI name.	None	JNDI name of the destination created in the JMS provider. This is the target destination for dead messages. This is used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
<code>DeadMessageDestinationClassName</code>	class name of destination object.	None	Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
<code>DeadMessageDestinationProperties</code>	Name Value Pairs separated by comma	None	Specifies the <code>javabean</code> property names and values of the destination of the JMS client. This is required only if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
<code>DeadMessageConnectionFactoryJndiName</code>	a valid JNDI name	None	JNDI name of the connection factory created in the JMS provider. This is the target connection factory for dead messages. This is used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
<code>DeadMessageDestinationType</code>	queue or topic destination	None	The destination type for dead messages.
<code>ReconnectAttempts</code>	integer	0	Number of times a reconnect will be attempted in case exception listener catches an error on connection.
<code>ReconnectInterval</code>	time in seconds	0	Interval between reconnects.

## Using GenericJMSRA with WebLogic JMS

You can configure GenericJMSRA to enable applications running in Eclipse GlassFish to send messages to, and receive messages from, Oracle WebLogic JMS.

GenericJMSRA should be used in conjunction with the WebLogic Server Thin T3 Client. Due to the nature of this client, messages exchanged between Eclipse GlassFish and WebLogic Server cannot be sent or received in XA transactions. There is also only limited support for asynchronous receipt of messages in an MDB, as described in detail in [Limitations When Using GenericJMSRA with WebLogic JMS](#).

The following topics are addressed here:

- [Deploy the WebLogic Thin T3 Client JAR in Eclipse GlassFish](#)
- [Configure WebLogic JMS Resources for Integration](#)
- [Create a Resource Adapter Configuration for GenericJMSRA to Work With WebLogic JMS](#)
- [Deploy the GenericJMSRA Resource Archive](#)
- [Configuring an MDB to Receive Messages from WebLogic JMS](#)
- [Accessing Connections and Destinations Directly](#)
- [Limitations When Using GenericJMSRA with WebLogic JMS](#)
- [Configuration Reference of GenericJMSRA Properties for WebLogic JMS](#)

### Deploy the WebLogic Thin T3 Client JAR in Eclipse GlassFish

WebLogic Server provides several different clients for use by standalone applications that run outside of WebLogic Server. These client are summarized in "[Overview of Stand-alone Clients](#)" in Programming Stand-alone Clients for Oracle WebLogic Server. When connecting from Eclipse GlassFish to WebLogic JMS resources you must use the WebLogic Thin T3 client, `wlthint3client.jar`.

There are a couple of methods to deploy the WebLogic Thin T3 client in Eclipse GlassFish and make it available to GenericJMSRA:

- To make the Thin T3 client available to all applications, copy the `wlthint3client.jar` to the `as-install/lib` directory under your Eclipse GlassFish installation. The Thin T3 client can be found in a WebLogic Server installation in a directory similar to `MW_HOME/server/lib`.
- It is also possible to deploy the Thin T3 client in a less global manner, so that it is specific to an individual application. For information on how to do this, see "[Application-Specific Class Loading](#)" in Eclipse GlassFish Application Development Guide.

### Configure WebLogic JMS Resources for Integration

If you need to configure the necessary WebLogic JMS resources on the WebLogic Server from which you want to access messages using Eclipse GlassFish, then follow the instructions in the WebLogic

Server documentation for configuring the necessary resources, such as destinations, and connection factories.

- JMS System Module Configuration
- Queue and Topic Destination Configuration
- Connection Factory Configuration

The example code snippets in this section refer to a WebLogic JMS connection factory named `WLoutboundQueueFactory` and queue destination named `WLoutboundQueue`. For conceptual overviews on configuring WebLogic JMS resources, refer to "[Understanding JMS Resource Configuration](#)" in Configuring and Managing JMS for Oracle WebLogic Server. For detailed instructions on configuring WebLogic JMS resources, refer to "[Configure JMS system modules and add JMS resources](#)" in the WebLogic Administration Console Online Help.

## Create a Resource Adapter Configuration for GenericJMSRA to Work With WebLogic JMS

When you deploy GenericJMSRA, you also need to create a resource adapter configuration in Eclipse GlassFish. You can do this using either the Administration Console or the `asadmin` command. If you use the Administration Console then you need to deploy the GenericJMSRA resource archive first. Here's an example using `asadmin`:

```
asadmin create-resource-adapter-config --host localhost --port 4848 --property  
SupportsXA=false:DeliveryType=Synchronous:ProviderIntegrationMode  
=jndi:JndiProperties=java.naming.factory.initial\  
  
=weblogic.jndi.WLInitialContextFactory,java.naming.provider.url\  
=t3://localhost:7001,java.naming.factory.url.pkgs\  
=weblogic.corba.client.naming genericra
```

This creates a resource adapter configuration with the name `genericra`, and Oracle recommends not changing the default name. The resource adapter configuration is configured with the properties specified using the `--properties` argument; multiple properties are configured as a colon-separated list of name-value pairs that are entered as a single line. You will also need to change the host and port that WebLogic Server is running on to suit your installation.

In this example, the following properties are configured:

Property	Value
<code>SupportsXA</code>	<code>false</code>
<code>DeliveryType</code>	<code>Synchronous</code>
<code>ProviderIntegrationMode</code>	<code>jndi</code>

Property	Value
JndiProperties	<pre>java.naming.factory.initial =weblogic.jndi.WLInitialContextFactory,java.naming.provider.url =t3://localhost:7001,java.naming.factory.url.pkgs =weblogic.corba.client.naming  (replace "localhost:7001" with the host:port of WebLogic Server)</pre>

You must use the same values for `SupportsXA`, `DeliveryType` and `ProviderIntegrationMode` as the required values that are used in this table. The `JndiProperties` value must be set to a list of JNDI properties needed for connecting to WebLogic JNDI.



When using `asadmin` you need to escape each `=` and any `:` characters by prepending a backward slash `\`. The escape sequence is not necessary if the configuration is performed through the Administration Console.

For a description of all the resource adapter properties that are relevant for WebLogic JMS, see the [Configuration Reference of GenericJMSRA Properties for WebLogic JMS](#).

## Deploy the GenericJMSRA Resource Archive

1. Download the GenericJMSRA resource archive (`genericra.rar`).
2. Deploy the resource adapter. You can do this using either the Administration Console or the `asadmin` deploy command. Here's an example using the `asadmin` deploy command:

```
$ asadmin deploy --user admin --password adminadmin
<location of the generic resource adapter rar file>
```

If you deploy the resource adapter using the Administration Console, then after deployment you need to create a resource adapter configuration as described in [Create a Resource Adapter Configuration for GenericJMSRA to Work With WebLogic JMS](#).

## Configuring an MDB to Receive Messages from WebLogic JMS

In this example, all configuration information is defined in two deployment descriptor files: `ejb-jar.xml` and the Eclipse GlassFish `glassfish-ejb-jar.xml` file. To configure a MDB to receive messages from WebLogic JMS, you would configure these deployment descriptor files as follows:

1. Configure the `ejb-jar.xml` deployment descriptor:

```

<ejb-jar>
  <enterprise-beans>
    <message-driven>
      <ejb-name>SimpleMessageEJB</ejb-name>
      <ejb-class>test.simple.queue.ejb.SimpleMessageBean</ejb-class>
      <transaction-type>Container</transaction-type>
    </message-driven>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>SimpleMessageEJB</ejb-name>
        <method-name>onMessage</method-name>
        <method-params>
          <method-param>jakarta.jms.Message</method-param>
        </method-params>
      </method>
      <trans-attribute>NotSupported</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>

```



If container-managed transactions are configured, then the transactional attribute must be set to `NotSupported`. For more information, see [Limitations When Using GenericJMSRA with WebLogic JMS](#).

## 2. Configure the glassfish-ejb-jar.xml deployment descriptor:

```

<sun-ejb-jar>
  <enterprise-beans>
    <ejb>
      <ejb-name>SimpleMessageEJB</ejb-name>
      <mdb-resource-adapter>
        <resource-adapter-mid>genericra</resource-adapter-mid>
        <activation-config>
          <activation-config-property>
            <activation-config-property-name>
              ConnectionFactoryJndiName
            </activation-config-property-name>
            <activation-config-property-value>
              jms/WLInboundQueueFactory
            </activation-config-property-value>
          </activation-config-property>
          <activation-config-property>
            <activation-config-property-name>
              DestinationJndiName
            </activation-config-property-name>
            <activation-config-property-value>

```

```
jms/WLInboundQueue
  </activation-config-property-value>
  </activation-config-property>
  </activation-config>
  </mdb-resource-adapter>
</ejb>
</enterprise-beans>
</sun-ejb-jar>
```

where:

The `<resource-adapter-mid>genericra</resource-adapter-mid>` element is used to specify the resource adapter and resource adapter configurations that was deployed in the [Create a Resource Adapter Configuration for GenericJMSRA to Work With WebLogic JMS](#) instructions. It is recommended you stick to `genericra` as is used here.

The `activation-config` element in `glassfish-ejb-jar.xml` is the one which defines how and where the MDB receives messages, as follows:

- \* The `ConnectionFactoryJndiName` property must be set to the JNDI name of the connection factory in the WebLogic JNDI store that will be used to receive messages. Therefore, replace `jms/WLInboundQueueFactory` in the example above with the JNDI name used in your environment.
- \* The `DestinationJndiName` property must be set to the JNDI name of the destination (the queue or topic from which messages will be consumed) in the WebLogic JNDI store. Therefore, replace `jms/WLInboundQueue` in the example above with the JNDI name used in your environment.

For a description of all the ActivationSpec properties that are relevant for WebLogic JMS, see the [Configuration Reference of GenericJMSRA Properties for WebLogic JMS](#).

Make sure to use the appropriate WebLogic administration tools, such as the WebLogic Administration Console or the WebLogic Scripting Tool (WLST). For more information, see ["Configure Messaging"](#) in WebLogic Server Administration Console Online Help and [WebLogic Server WLST Online and Offline Command Reference](#).

## Accessing Connections and Destinations Directly

When configuring a MDB to consume messages from WebLogic JMS your code does not need to access the WebLogic JMS connection factory and destination directly. You simply define them in the activation configuration, as shown in [Configuring an MDB to Receive Messages from WebLogic JMS](#). However when configuring an MDB to send messages, or when configuring a EJB, Servlet, or application client to either send or receive messages, your code needs to obtain these objects using a JNDI lookup.

 If you want configure connections and destination resources using the Administration Console, this is explained in the Administration Console online help. When using Administration Console, follow the instructions for creating a new Connector Connection Pool and Admin Object Resources, and not the instructions for creating a JMS Connection Pool and Destination Resources. For

more information about using `asadmin` to create these resources, see [To Create a Connector Connection Pool](#) and [To Create a Connector Resource](#).

## 1. Looking up the connection factory and destination

The following code looks up a connection factory with the JNDI name `jms/QCFactory` and a queue with the name `jms/outboundQueue` from the Eclipse GlassFish JNDI store:

```
Context initialContext = new InitialContext();
QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
    jndiContext.lookup("java:comp/env/jms/MyQCFactory");
Queue queue = (Queue) jndiContext.lookup("java:comp/env/jms/outboundQueue");
```

Note that the resources used are Eclipse GlassFish resources, not WebLogic JMS resources. For every connection factory or destination that you want to use in the WebLogic JMS JNDI store, you need to create a corresponding connection factory or destination in the Eclipse GlassFish JNDI store and configure the Eclipse GlassFish object to point to the corresponding WebLogic JMS object.

## 2. Declaring the connection factory and destination

In accordance with standard Jakarta EE requirements, these resources need to be declared in the deployment descriptor for the MDB, EJB or other component. For example, for a session bean, configure the `ejb-jar.xml` with `<resource-env-ref>` elements, as follows:

```
<ejb-jar>
  <enterprise-beans>
    <session>
      ...
      <resource-env-ref>
        <resource-env-ref-name>jms/QCFactory</resource-env-ref-name>
        <resource-env-ref-type>jakarta.jms.QueueConnectionFactory</resource-env-
ref-type>
      </resource-env-ref>
      <resource-env-ref>
        <resource-env-ref-name>jms/outboundQueue</resource-env-ref-name>
        <resource-env-ref-type>jakarta.jms.Queue</resource-env-ref-type>
      </resource-env-ref>
    </session>
  </enterprise-beans>
</ejb-jar>
```

## 3. Create a Connector Connection Pool and Connector Resource by entering the following `asadmin` commands, both all in one line:

In order to configure a JMS Connection Factory using GenericJMSRA, a Connector connection pool and resource need to be created in Eclipse GlassFish using names that map to the corresponding connection factory in the WebLogic JNDI store.

```
asadmin create-connector-connection-pool --host localhost --port 4848
--rname genericra --connectiondefinition jakarta.jms.QueueConnectionFactory
```

```
--target server --transactionsupport LocalTransaction
--property ConnectionFactoryJndiName=jms/WLOutboundQueueFactory
    qcpool

asadmin create-connector-resource --host localhost --port 4848
    --poolname qcpool --target server jms/QCFactory
```

These `asadmin` commands together creates a connection factory in Eclipse GlassFish and its corresponding connection pool.

- The connection pool has the JNDI name `jms/WLoutboundQueueFactory` and obtains connections from a connection pool named `qcpool`.
- The connection pool `qcpool` uses the resource adapter `genericra` and contains objects of type `jakarta.jms.QueueConnectionFactory`.
- The `transactionsupport` argument is set to `LocalTransaction`, which specifies that the connection will be used in local transactions only. You can also specify `NoTransaction`. However, the default setting of `XATransaction` cannot be used. For more information, see [Limitations When Using GenericJMSRA with WebLogic JMS](#).
- The connection pool is configured with the properties specified using the `properties` argument; multiple properties are configured as a colon-separated list of name-value pairs. Only one property is configured in this example, as follows:

`ConnectionFactoryJndiName=jms/WLOutboundQueueFactory`

The `ConnectionFactoryJndiName` property must be set to the JNDI name of the corresponding connection factory in the WebLogic JMS JNDI store. Therefore, replace `jms/WLOutboundQueueFactory` in the example above with the JNDI name used in your environment.

- For a description of the `ManagedConnectionFactory` properties that are relevant for WebLogic JMS, see the [Configuration Reference of GenericJMSRA Properties for WebLogic JMS](#).
4. Create a destination object that refers to a corresponding WebLogic JMS destination by entering the following `asadmin` command, all in one line:

```
asadmin create-admin-object --host localhost --port 4848 --target server
    --restype jakarta.jms.Queue --property DestinationJndiName=jms/WLOutboundQueue
    --rname genericra jms/outboundQueue
```

This `asadmin` command creates a destination in Eclipse GlassFish.

- The destination has the JNDI name `jms/outboundQueue`, uses the resource adapter `genericra`, and is of type `jakarta.jms.Queue`.
- The destination is configured with the properties specified using the `properties` argument; multiple properties are configured as a colon-separated list of name-value pairs. Only one property is configured in this example, as follows:

## DestinationJndiName=jms/WLOutboundQueue

The `DestinationJndiName` property must be set to the JNDI name of the corresponding destination in the WebLogic JMS JNDI store. Therefore, replace `jms/WLOutboundQueue` in the example above with the JNDI name used in your environment.

- For a description of the destination properties that are relevant for WebLogic JMS, see the [Configuration Reference of GenericJMSRA Properties for WebLogic JMS](#).

## Limitations When Using GenericJMSRA with WebLogic JMS

Due to the nature of the WebLogic T3 Thin Client there are a number of limitations in the way in which it can be used with GenericJMSRA.

### No Support for XA Transactions

WebLogic JMS does not support the optional JMS "Chapter 8" interfaces for XA transactions in a form suitable for use outside of WebLogic Server. Therefore, the GenericJMSRA configuration must have the `SupportsXA` property set to `-false`. This has a number of implications for the way in which applications may be used, as described in this section.

#### Using a MDB to Receive Messages: Container-managed Transactions (CMT)

- If container-managed transactions are used, the transactional attribute of a MDB should be set to `NotSupported`. No transaction will be started. Messages will be received in a non-transacted session with an acknowledgeMode of `AUTO_ACKNOWLEDGE`.
- A transactional `Required` attribute should not be used; otherwise, MDB activation will fail with an exception: `javax.resource.ResourceException: MDB is configured to use container managed transaction. But SupportsXA is configured to false in the resource adapter.`

The remaining transactional attributes are normally considered inappropriate for use with a MDB. If used, the following behavior will occur:

- If the transactional attribute is `RequiresNew`, then MDB activation will fail with an exception: `javax.resource.ResourceException: MDB is configured to use container managed transaction. But SupportsXA is configured to false in the resource adapter.`
- If the transactional attribute is `Mandatory`, the MDB can be activated but a `TransactionRequiredException` (or similar) will always be thrown by the server.
- If the transactional attribute is `Supports`, then no transaction will be started and the MDB will work as if `NotSupported` had been used.
- If the transactional attribute is `Never`, then no transaction will be started and the MDB will work as if `NotSupported` had been used.

#### Using a MDB to Receive Messages: Bean-managed Transactions (BMT)

- If bean-managed transactions are configured in accordance with the EJB specification any `UserTransaction` started by the bean will have no effect on the consumption of messages.
- Messages will be received in a non-transacted session with an `acknowledgeMode` of `AUTO_ACKNOWLEDGE`.

## Accessing Connections and Destinations Directly - Container-managed Transactions (CMT)

When accessing connections directly (such as when sending messages from a MDB or an EJB) and container-managed transactions are being used, the connection pool's `transaction-support` property should be set to either `LocalTransaction` or `NoTransaction`. If the default value of `XATransaction` is used, an exception will be thrown at runtime when `createConnection()` is called. This is the case irrespective of the transactional attribute of the MDB or EJB. Note that MDBs must have their transactional attribute set to `NotSupported` as specified above; whereas, an EJB can use any transactional attribute.

If there is no transaction in progress within the bean method (for example, `notSupported` is being used) then it does not make any difference whether the connection pool's `transaction-support` property is set to `LocalTransaction` or `NoTransaction`; the transactional behavior will be determined by the arguments to `createSession()`. If you want the outbound message to be sent without a transaction, call `createSession(false, ...)`. If you want the outbound message to be sent in a local transaction call `createSession(true, Session.SESSION_TRANSACTED)`, remembering to call `session.commit()` or `session.rollback()` after the message is sent.

If there is a transaction in progress within the bean method (which will only be possible for EJBs), then setting the connection pool's `transaction-support` property to `LocalTransaction` or `NoTransaction` gives different results:

- If it is set to `NoTransaction` then a non-transacted session will be used.
- If it is set to `LocalTransaction` then a (local, non-XA) transacted session will be used, which will be committed or rolled back when the `UserTransaction` is committed or rolled back. In this case, calling `session.commit()` or `session.rollback()` will cause an exception.

## No Support for Redelivery Limits and Dead Message Queue

Due to the lack of XA support when using WebLogic JMS, there is no support for GenericJMSRA's dead message queue feature, in which a message that has been redelivered to a MDB a defined number of times is sent to a dead message queue.

## Limited Support for Asynchronous Receipt of Messages In a MDB

WebLogic JMS does not support the optional JMS "Chapter 8" interfaces for "Concurrent Processing of a Subscription's Messages" (that is, `ServerSession`, `ServerSessionPool` and `ConnectionConsumer`) in a form suitable for use outside of WebLogic Server. Therefore, the generic JMSRA configuration must set the property `DeliveryType` to `Synchronous`.

This affects the way in which MDBs consume messages from a queue or topic as follows:

- When messages are being received from a queue, each MDB instance will have its own session

and consumer, and it will consume messages by repeatedly calling `receive(timeout)`. This allows the use of a pool of MDBs to process messages from the queue.

- When messages are being received from a topic, only one MDB instance will be used irrespective of the configured pool size. This means that a pool of multiple MDBs cannot be used to share the load of processing messages, which may reduce the rate at which messages can be received and processed.

This restriction is a consequence of the semantics of synchronously consuming messages from topics in JMS: In the case of non-durable topic subscriptions, each consumer receives a copy of all the messages on the topic, so using multiple consumers would result in multiple copies of each message being received rather than allowing the load to be shared among the multiple MDBs. In the case of durable topic subscriptions, only one active consumer is allowed to exist at a time.

## Configuration Reference of GenericJMSRA Properties for WebLogic JMS

The tables in this section list the properties that need to be set to configure the resource adapter and any activation specs, managed connections, and other administered objects that are relevant only when using GenericJMSRA to communicate with WebLogic JMS. For a complete list of properties, see the comprehensive table in [GenericJMSRA Configuration Properties](#).

### Resource Adapter Properties

These properties are used to configure the resource adapter itself when it is deployed, and can be specified using the `create-resource-adapter-config` command.

Property Name	Required Value	Description
<code>SupportsXA</code>	<code>false</code>	Specifies whether the JMS client supports XA transactions. Set to <code>false</code> for WebLogic JMS.
<code>DeliveryType</code>	<code>Synchronous</code>	Specifies whether an MDB should use a <code>ConnectionConsumer</code> (Asynchronous) or <code>consumer.receive()</code> (Synchronous) when consuming messages. Set to <code>Synchronous</code> for WebLogic JMS.

Property Name	Required Value	Description
<code>ProviderIntegrationMode</code>	<code>jndi</code>	<p>Specifies that connection factories and destinations in GlassFish's JNDI store are configured to refer to connection factories and destinations in WebLogic's JNDI store.</p> <p>Set to <code>jndi</code> for WebLogic JMS.</p>
<code>JndiProperties</code>	<pre>java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory, java.naming.provider.url=t3://localhost:7001, java.naming.factory.url.pkgs=weblogic.corba.client.naming (replace localhost:7001 with the host:port of WebLogic Server)</pre>	JNDI properties for connect to WebLogic JNDI, specified as comma-separated list of name=value pairs without spaces.
<code>UserName</code>	Name of the WebLogic JMS user	User name to connect to WebLogic JMS. The user name can be overridden in <code>ActivationSpec</code> and <code>ManagedConnection</code> . If no user name is specified anonymous connections will be used, if permitted.
<code>Password</code>	Password for the WebLogic JMS user	Password to connect to WebLogic JMS. The password can be overridden in <code>ActivationSpec</code> and <code>ManagedConnection</code> .
<code>LogLevel</code>	Desired log level of JDK logger	Used to specify the level of logging.

## Connection Factory Properties

`ManagedConnectionFactory` objects are created in the Eclipse GlassFish JNDI store using the Administration Console or the `asadmin connector-connection-pool` command. All the properties that can be set on a resource adapter configuration can be overridden by setting them on a destination object. The properties specific to `ManagedConnectionFactory` objects are listed in the following table.

Property Name	Valid Value	Default Value	Description
<code>ClientId</code>	A valid client ID	None	<code>ClientID</code> as specified by JMS 1.1 specification.
<code>ConnectionFactoryJndiName</code>	A valid JNDI Name	None	JNDI name of connection factory in the Eclipse GlassFish JNDI store. This connection factory should be configured to refer to the physical connection factory in the WebLogic JNDI store.
<code>ConnectionValidationEnabled</code>	<code>true</code> or <code>false</code>	FALSE	If set to <code>true</code> , the resource adapter will use an exception listener to catch any connection exception and will send a <code>CONNECTION_ERROR_OCCURRED</code> event to Eclipse GlassFish.

## Destination Properties

Destination (queue or topic) objects are created in the Eclipse GlassFish JNDI store using the Administration Console or the `asadmin connector-admin-object` command. All the properties that can be set on a resource adapter configuration can be overridden by setting them on a destination object. The properties specific to destination objects are listed in the following table.

Property Name	Valid Value	Default Value	Description
<code>DestinationJndiName</code>	A valid JNDI name	None	JNDI name of the destination object in the Eclipse GlassFish JNDI store. This destination object should be configured to refer to the corresponding physical destination in the WebLogic JNDI store.

## ActivationSpec Properties

An ActivationSpec is a set of properties that configures a MDB. It is defined either in the MDB's Eclipse GlassFish deployment descriptor `glassfish-ejb-jar.xml` using `activation-config-property` elements or in the MDB itself using annotation. All the resource adapter properties listed in the table above can be overridden in an ActivationSpec. Additional properties available only to a ActivationSpec are given below.

Property Name	Valid Value	Default Value	Description
<code>MaxPoolSize</code>	An integer	8	<p>Maximum size of server session pool internally created by the resource adapter for achieving concurrent message delivery. This should be equal to the maximum pool size of MDB objects.</p> <p>Only used for queues; ignored for topics, when a value of 1 is always used.</p>
<code>SubscriptionDurability</code>	Durable or Non-Durable	Non-Durable	Only used for topics. Specifies whether the subscription is durable or non-durable.
<code>SubscriptionName</code>		None	Only used for topics when <code>SubscriptionDurability</code> is Durable. Specifies the name of the durable subscription.
<code>MessageSelector</code>	A valid message selector	None	JMS message selector.
<code>ClientID</code>	A valid client ID	None	JMS ClientID.
<code>ConnectionFactoryJndiName</code>	A valid JNDI Name	None	JNDI name of connection factory in the Eclipse GlassFish JNDI store. This connection factory should be configured to refer to the physical connection factory in the WebLogic JNDI store.
<code>DestinationJndiName</code>	A valid JNDI Name	None	JNDI name of destination in the Eclipse GlassFish JNDI store. This destination should be configured to refer to the physical destination in the WebLogic JNDI store.
<code>DestinationType</code>	<code>jakarta.jms.Queue</code> or <code>jakarta.jms.Topic</code>	Null	Specifies whether the configured <code>DestinationJndiName</code> refers to a queue or topic.
<code>ReconnectAttempts</code>	integer	0	Number of times a reconnect will be attempted in case exception listener catches an error on connection.
<code>ReconnectInterval</code>	time in seconds	0	Interval between reconnection attempts.

## Using GenericJMSRA with IBM WebSphere MQ

You can configure GenericJMSRA to enable applications running in Eclipse GlassFish to send messages to, and receive messages from, IBM WebSphere MQ. Eclipse GlassFish only supports using GenericJMSRA with WebSphere MQ version 6.0 and WebSphere MQ version 7.0.

These instructions assume that the WebSphere MQ broker and Eclipse GlassFish are deployed and running on the same physical host/machine. If you have the WebSphere MQ broker running on a different machine and need to access it remotely, refer to the WebSphere MQ documentation for configuration details. The resource adapter configuration and other application server related configuration remains unchanged.

The following topics are addressed here:

- [Preliminary Setup Procedures for WebSphere MQ Integration](#)
- [Configure the WebSphere MQ Administered Objects](#)
- [Create a Resource Adapter Configuration for GenericJMSRA to Work With WebSphere MQ](#)
- [Deploy the GenericJMSRA Archive](#)
- [Create the Connection Factories and Administered Objects in Eclipse GlassFish](#)
- [Configuring an MDB to Receive Messages from WebSphere MQ](#)

### Preliminary Setup Procedures for WebSphere MQ Integration

Before you can configure WebSphere MQ to exchange messages with Eclipse GlassFish, you must complete the following tasks:

- The following permissions must be added to the `server.policy` and the `client.policy` file to deploy GenericJMSRA and to run the client application.
  - Use a text editor to modify the `server.policy` file in the `\${appserver-install-dir}/domains/domain1/config/` directory by adding the following line to the default grant block:

```
permission java.util.logging.LoggingPermission "control";
permission java.util.PropertyPermission "*", "read,write";
```

- If you use an application client in your application, edit the client's `client.policy` file in the `${appserver-install-dir}/lib/appclient/` directory by adding the following permission:

```
permission javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential * \\"*\\"", "read";
```

- To integrate Eclipse GlassFish with WebSphere MQ 6.0 or 7.0, copy the necessary JAR files to the `as-install/lib` directory:

- For WebSphere MQ 6.0, copy these JAR files to the as-install/**lib** directory:

```
/opt/mqm/java/lib/com.ibm.mq.jar  
/opt/mqm/java/lib/com.ibm.mq.jms.Nojndi.jar  
/opt/mqm/java/lib/com.ibm.mq.soap.jar  
/opt/mqm/java/lib/com.ibm.mqjms.jar  
/opt/mqm/java/lib/com.ibm.mqetclient.jar  
/opt/mqm/java/lib/commonservices.jar  
/opt/mqm/java/lib/dhbcore.jar  
/opt/mqm/java/lib/rmm.jar  
/opt/mqm/java/lib/providerutil.jar  
/opt/mqm/java/lib/CL3Export.jar  
/opt/mqm/java/lib/CL3Nonexport.jar
```

where **/opt/mqm** is the location of the WebSphere MQ 6.0 installation.

- For WebSphere MQ 7.0, copy these JAR files to the as-install/**lib** directory:

```
/opt/mqm/java/lib/com.ibm.mq.jar,  
/opt/mqm/java/lib/com.ibm.mq.jms.Nojndi.jar,  
/opt/mqm/java/lib/com.ibm.mq.soap.jar,  
/opt/mqm/java/lib/com.ibm.mqjms.jar,  
/opt/mqm/java/lib/com.ibm.mq.jmqi.jar,  
/opt/mqm/java/lib/com.ibm.mq.commonservices.jar,  
/opt/mqm/java/lib/dhbcore.jar,  
/opt/mqm/java/lib/rmm.jar,  
/opt/mqm/java/lib/providerutil.jar,  
/opt/mqm/java/lib/CL3Export.jar,  
/opt/mqm/java/lib/CL3Nonexport.jar
```

where **/opt/mqm** is the location of the WebSphere MQ 7.0 installation.

- Set the **LD\_LIBRARY\_PATH** environment variable to the **java/lib** directory, and then restart Eclipse GlassFish. For example, in a UNIX—based system, with WebSphere MQ installed under **/opt/mqm**, you would enter:

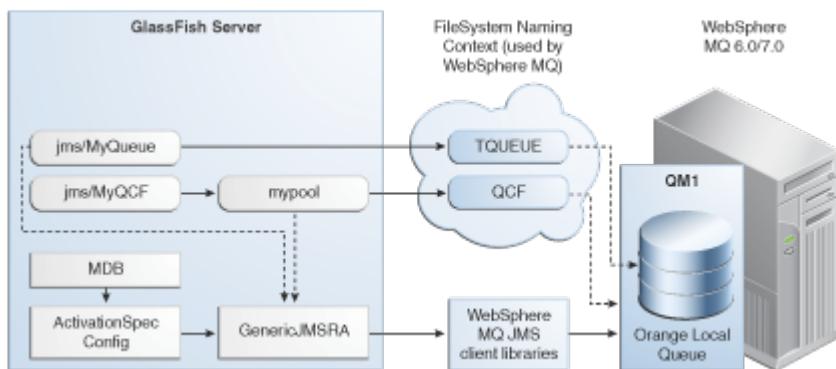
```
$ export LD_LIBRARY_PATH=/opt/mqm/java/lib
```

## Configure the WebSphere MQ Administered Objects

This section provides an example of how you could configure the necessary administered objects, such as destinations and connection factories, on the WebSphere MQ instance from which you want to access messages using Eclipse GlassFish. Therefore, you will need to change the administered object names to suit your installation.

## Before You Begin

If WebSphere MQ created a user and a group named `mqm` during the installation, then you must specify a password for the `mqm` user using the `$ passwd mqm` command.



1. Switch to the `mqm` user:

```
$ su mqm
```

2. For Linux, set the following kernel version:

```
$ export LD_ASSUME_KERNEL=2.2.5
```

3. Create a new MQ queue manager named "QM1":

```
$ crtqm QM1
```

4. Start the new MQ queue manager.

In the image above, `QM1` is associated with the IBM WebSphere MQ broker.

```
$ strmqm QM1
```

5. Start the MQ listener:

```
$ runmqlsr -t tcp -m QM1 -p 1414 &
```

6. Modify the default JMSAdmin console configuration as follows:

1. Edit the JMSAdmin script in the `/opt/mqm/java/bin` directory to change the JVM to a location of a valid JVM your system.
2. Set the relevant environment variable required for JMSAdmin by sourcing the `setjmsenv` script located in the `/opt/mqm/java/bin` directory.

```
$ cd /opt/mqm/java/bin  
$ source setjmsenv
```

where `/opt/mqm` is the location of the WebSphere MQ installation.

3. Change the `JMSAdmin.config` file to indicate the Initial Context Factory you will be using by setting the following name-value pairs and commenting out the rest:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory  
PROVIDER_URL=file:/opt/tmp
```

7. Create WebSphere MQ queues using the `runmqsc` console and `MQJMS_PSQ.mqsc` script.

```
$ runmqsc QM1 < MQJMS_PSQ.mqsc
```

8. Create user defined physical queue for your application using runmqsc console and an appropriate physical queue name. An example of how this could be done is shown below. In the image above, **ORANGE.LOCAL.QUEUE** is associated with **QM1**.

```
$ runmqsc QM1  
> DEFINE QLOCAL(ORANGE.LOCAL.QUEUE)  
> end
```

9. Start the WebSphere MQ Broker:

```
$ strmqbrk -m QM1
```

10. In the WebSphere MQ JMSAdmin console, use the following commands to create the connection factories, XA connection factories, and destinations for your application, as shown in the following sample, which lists each of the various JMS administered objects.

In the image above, **QCF** (for **QM1**) and **TQueue** (associated with **ORANGE.LOCAL.QUEUE**) are defined in the **FileSystem Naming Context**.

```
$ ./JMSAdmin
```

```
InitCtx>def qcf<JNDI name to be given to the Queue Connection Factory>  
hostname<IBM MQ server hostname> port(1414) channel(SYSTEM.DEF.SVRCONN)  
transport(CLIENT) qmanager<name of queue manager defined>
```

For example:

```
def qcf(QCF) hostname(localhost) port(1414) channel(SYSTEM.DEF.SVRCONN)  
transport(CLIENT) qmanager(QM1)
```

```
InitCtx%def xaqcf<JNDI name to be given to the XA Queue Connection Factory>  
hostname<IBM MQ server hostname> port(1414) channel(SYSTEM.DEF.SVRCONN)  
transport(CLIENT) qmanager<name of queue manager defined>
```

For example:

```
def xaqcf(XAQCF) hostname(localhost) port(1414) channel(SYSTEM.DEF.SVRCONN)  
transport(CLIENT) qmanager(QM1)
```

```
InitCtx%def q<JNDI Name to be given to the Queue> queue<physical queue name>  
qmanager<name of queue manager defined >
```

For example: def q(TQueue) queue(ORANGE.LOCAL.QUEUE) qmanager(QM1)

```
InitCtx%def tcf<JNDI Name to be given to the Topic Connection Factory>  
qmanager<name of queue manager defined >
```

For example: def tcf(TCF) qmanager(QM1)

```
InitCtx%def xatcf<JNDI Name to be given to the XA Topic Connection Factory>  
qmanager<name of queue manager defined >
```

For example: def xatcf(XATCF) qmanager(QM1)

InitCtx%def t<JNDI Name to be given to the Topic> topic<sample topic name>

For example: def t(TTopic) topic(topic)

## Create a Resource Adapter Configuration for GenericJMSRA to Work With WebSphere MQ

Before deploying GenericJMSRA, you need to create a resource adapter configuration in Eclipse GlassFish. You can do this using either the Administration Console or the `asadmin` command. Use the following `asadmin` command to create a resource adapter configuration for `genericra` to configure it to work with WebSphere MQ.

```
asadmin> create-resource-adapter-config
--user <adminname> --password <admin password>
--property SupportsXA=true:ProviderIntegrationMode
=jndi:UserName=mqm:Password=###:RMPolicy
=OnePerPhysicalConnection:JndiProperties
=java.naming.factory.url.pkgs\\
=com.ibm.mq.jms.naming,java.naming.factory.initial\\
=com.sun.jndi.fscontext.RefFSContextFactory,java.naming.provider.url\\
=file\\:\\\\opt\\tmp:LogLevel=finest genericra
```

 When using `asadmin` you need to escape each `=` and any `:` characters by prepending a backward slash `\`. The escape sequence is not necessary if the configuration is performed through the Administration Console. Also , ensure that the provider URL is configured correctly depending on the platform. For example, on Windows systems it should be `file:/C:/opt/tmp` and on UNIX-based systems it is `file://opt/tmp`.

This creates a resource adapter configuration with the name `genericra`, and Oracle recommends not changing the default name. The resource adapter configuration is configured with the properties specified using the `--properties` argument; multiple properties are configured as a colon-separated list of name-value pairs that are entered as a single line.

In this example, the following properties are configured:

 The tables in this section describe the GenericJMSRA properties that are relevant only when integrating with WebSphere MQ. For a complete list of properties, see the comprehensive table in [GenericJMSRA Configuration Properties](#).

Property Name	Required Value	Description
<code>SupportsXA</code>	<code>true</code>	Set the supports distributed transactions attribute to <code>true</code> . The level of transactional support the adapter provides—none, local, or XA—depends on the capabilities of the Enterprise Information System (EIS) being adapted. If an adapter supports XA transactions and this attribute is XA, the application can use distributed transactions to coordinate the EIS resource with JDBC and JMS resources.
<code>ProviderIntegrationMode</code>	<code>jndi</code>	Specifies that connection factories and destinations in GlassFish's JNDI store are configured to refer to connection factories and destinations in WebSphere MQ's JNDI store.
<code>JndiProperties</code>	<pre>JndiProperties=java.naming.factory.url.pkgs\ =com.ibm.mq.jms.naming, java.naming.factory.initial\ =com.sun.jndi.fscontext.RefFSContextFactory, java.naming.provider.url\ =file\ :\ \\opt\ tmp:LogLevel=finest genericra</pre>	JNDI properties for connecting to WebSphere MQ's JNDI, specified as comma-separated list of name=value pairs without spaces.
<code>UserName</code>	Name of the WebSphere MQ user	User name to connect to WebSphere MQ. The user name can be overridden in <code>ActivationSpec</code> and <code>ManagedConnection</code> . If no user name is specified anonymous connections will be used, if permitted.

Property Name	Required Value	Description
Password	Password for the WebSphere MQ user	Password to connect to WebSphere MQ. The password can be overridden in ActivationSpec and ManagedConnection.
RMPolicy	OnePerPhysicalConnection	<p>Some XAResource implementations, such as WebSphere MQ, rely on a Resource Manager per Physical Connection, and this causes issues when there is inbound and outbound communication to the same queue manager in a single transaction (for example, when an MDB sends a response to a destination).</p> <p>When RMPolicy is set to OnePerPhysicalConnection, the XAResource wrapper implementation's <code>isSameRM</code> in GenericJMSRA would check if both the XAResources use the same physical connection, before delegating to the wrapped objects. Therefore, ensure that this attribute is set to OnePerPhysicalConnection if the application uses XA.</p>
LogLevel	Desired log level of JDK logger	Used to specify the level of logging.



You must use the values for `SupportsXA`, `RMPolicy` and `ProviderIntegrationMode` as the required values that are used in this table.

## Deploy the GenericJMSRA Archive

For instructions on downloading and deploying GenericJMSRA, see [Deploy the GenericJMSRA Resource Archive](#).

## Create the Connection Factories and Administered Objects in Eclipse GlassFish

In order to configure a JMS Connection Factory using GenericJMSRA, a Connector Connection Pool and resource needs to be created in Eclipse GlassFish, as described in this section.

Using the example WebSphere MQ configuration in [Configure the WebSphere MQ Administered Objects](#), you will see `mypool` (pointing to `GenericJMSRA` and `QCF`) and `jms/MyQCF` (for `mypool`) created in Eclipse GlassFish.



If you want to configure connections and destination resources using the Administration Console, this is explained in the Administration Console online help. When using Administration Console, follow the instructions for creating a new Connector Connection Pool and Admin Object Resources, and not the instructions for creating a JMS Connection Pool and Destination Resources. For more information about using `asadmin` to create these resources, see [To Create a Connector Connection Pool](#) and [To Create a Connector Resource](#).

### Creating Connections and Destinations

In order to configure a JMS Connection Factory, using GenericJMSRA, a Connector Connection Pool and Destination resources need to be created in Eclipse GlassFish using names that map to the corresponding connection and destination resources in WebSphere MQ. The connections and destination name in these steps map to the example WebSphere MQ configuration in [Configure the WebSphere MQ Administered Objects](#).

1. Create connection pools that point to the connection pools in WebSphere MQ.

The following `asadmin` command creates a Connection Pool called `mypool` and points to the `XACF` created in WebSphere MQ:

```
asadmin create-connector-connection-pool -- rENAME genericra
connectiondefinition
jakarta.jms.QueueConnectionFactory --transactionSupport XATransaction
--property ConnectionFactoryJndiName=QCF mypool
```

The following `asadmin` command creates a Connection Pool called `mypool2` and points to the `XATCF` created in WebSphere MQ:

```
asadmin create-connector-connection-pool
-- rENAME genericra connectiondefinition
jakarta.jms.TopicConnectionFactory
--transactionSupport XATransaction
--property ConnectionFactoryJndiName=XATCF mypool2
```

2. Create the connector resources.

The following `asadmin` command creates a connector resource named `jms/MyQCF` and binds this resource to JNDI for applications to use:

```
asadmin create-connector-resource --poolname mypool jms/MyQCF
```

The following `asadmin` command creates a connector resource named `jms/MyTCF` and binds this resource to JNDI for applications to use:

```
asadmin create-connector-resource --poolname mypool2 jms/MyTCF
```

### 3. Create the JMS destination resources as administered objects.

In the image above, `jms/MyQueue` (pointing to `GenericJMSRA` and `TQueue`) is created in Eclipse GlassFish.

The following `asadmin` command creates a `jakarta.jms.Queue` administered object and binds it to the Eclipse GlassFish JNDI tree at `jms/MyQueue` and points to the `jms/TQueue` created in WebSphere MQ.

```
asadmin create-admin-object --rname genericra --restype jakarta.jms.Queue  
--property DestinationJndiName=TQueue jms/MyQueue
```

The following `asadmin` command creates a `jakarta.jms.Topic` administered object and binds it to the Eclipse GlassFish JNDI tree at `jms/MyTopic` and points to the `jms/TTopic` created in WebSphere MQ.

```
asadmin create-admin-object --rname genericra --restype jakarta.jms.Topic  
--property DestinationJndiName=TTopic jms/MyTopic
```

## Configuring an MDB to Receive Messages from WebSphere MQ

The administered object names in the sample deployment descriptor below map to the example WebSphere MQ configuration in [Configure the WebSphere MQ Administered Objects](#). The deployment descriptors need to take into account the resource adapter and the connection resources that have been created. A sample `sun-ejb-jar.xml` for a Message Driven Bean that listens to a destination called `TQueue` in WebSphere MQ, and publishes back reply messages to a destination resource named `jms/replyQueue` in Eclipse GlassFish, as shown below.

```
<sun-ejb-jar>  
  <enterprise-beans>  
    <unique-id.1</unique-id>  
    <ejb>  
      <ejb-name>SimpleMessageEJB</ejb-name>
```

```

<jndi-name>jms/SampleQueue</jndi-name>
<!-- QCF used to publish reply messages -->
<resource-ref>
    <res-ref-name>jms/MyQueueConnectionFactory</res-ref-name>
    <jndi-name>jms/MyQCF</jndi-name>
    <default-resource-principal>
        <name>mqm</name>
        <password>mqm</password>
    </default-resource-principal>
</resource-ref>
<!-- reply destination resource> Creating of this replyQueue destination
resource is not
shown above, but the steps are similar to creating the "jms/MyQueue"
resource -->
<resource-env-ref>
    <resource-env-ref-name>jms/replyQueue</resource-env-ref-name>
    <jndi-name>jms/replyQueue</jndi-name>
</resource-env-ref>

<!-- Activation related RA specific configuration for this MDB -->
<mdb-resource-adapter>
<!-- resource-adapter-mid points to the Generic Resource Adapter for JMS
-->
<resource-adapter-mid>genericra</resource-adapter-mid>
<activation-config>
    <activation-config-property>
        <activation-config-property-name>DestinationType</activation-config-
property-name>
        <activation-config-property-value>javax>jms>Queue</activation-
config-property-value>
    </activation-config-property>
    <activation-config-property>
        <activation-config-property-name>
ConnectionFactoryJndiName</activation-config-property-name>
        <activation-config-property-value>QCF</activation-config-property-
value>
    </activation-config-property>
    <activation-config-property>
        <activation-config-property-name>DestinationJndiName</activation-
config-property-name>
        <activation-config-property-value>TQueue</activation-config-
property-value>
    </activation-config-property>
    <activation-config-property>
        <activation-config-property-name>MaxPoolSize</activation-config-
property-name>
        <activation-config-property-value>32</activation-config-property-
value>
    </activation-config-property>
    <activation-config-property>
        <activation-config-property-name>RedeliveryAttempts</activation-

```

```

config-property-name>
    <activation-config-property-value>0</activation-config-property-
value>
    </activation-config-property>
    <activation-config-property>
        <activation-config-property-name>ReconnectAttempts</activation-
config-property-name>
            <activation-config-property-value>4</activation-config-property-
value>
            </activation-config-property>
            <activation-config-property>
                <activation-config-property-name>ReconnectInterval</activation-
config-property-name>
                    <activation-config-property-value>10</activation-config-property-
value>
                    </activation-config-property>
                    <activation-config-property>
                        <activation-config-property-name>RedeliveryInterval</activation-
config-property-name>
                            <activation-config-property-value>1</activation-config-property-
value>
                            </activation-config-property>
                            <activation-config-property>
                                <activation-config-property-name>SendBadMessagesToDMD</activation-
config-property-name>
                                    <activation-config-property-value>false</activation-config-property-
value>
                                    </activation-config-property>
                                </activation-config>
                            </mdb-resource-adapter>
                            </ejb>
                            </enterprise-beans>
                        </sun-ejb-jar>

```

The business logic encoded in Message Driven Bean could then lookup the configured **QueueConnectionFactory/Destination** resource to create a connection as shown below.

```

Context context = null;
ConnectionFactory connectionFactory = null;
logger.info("In PublisherBean>ejbCreate()");
try {
    context = new InitialContext();
    queue = (javax.jms.Queue) context.lookup (
"java:comp/env/jms/replyQueue");
    connectionFactory = (ConnectionFactory) context>lookup(
"java:comp/env/jms/MyQueueConnectionFactory");
    connection = connectionFactory>createConnection();
} catch (Throwable t) {
    logger.severe("PublisherBean>ejbCreate:" + "Exception: " + t.
toString());
}

```

}

# 18 Administering the Java Naming and Directory Interface (JNDI) Service

The Java Naming and Directory Interface (JNDI) API is used for accessing different kinds of naming and directory services. Jakarta EE components locate objects by invoking the JNDI lookup method.

The following topics are addressed here:

- [About JNDI](#)
- [Administering JNDI Resources](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

## About JNDI

By making calls to the JNDI API, applications locate resources and other program objects. A resource is a program object that provides connections to systems, such as database servers and messaging systems. A JDBC resource is sometimes referred to as a data source. Each resource object is identified by a unique, people-friendly name, called the JNDI name. A resource object and its JNDI name are bound together by the naming and directory service, which is included with the Eclipse GlassFish.

When a new name-object binding is entered into the JNDI, a new resource is created.

The following topics are addressed here:

- [Jakarta EE Naming Environment](#)
- [How the Naming Environment and the Container Work Together](#)
- [Naming References and Binding Information](#)

## Jakarta EE Naming Environment

JNDI names are bound to their objects by the naming and directory service that is provided by a Jakarta EE server. Because Jakarta EE components access this service through the JNDI API, the object usually uses its JNDI name. For example, the JNDI name of the Apache Derby database is `jdbc/_default`. At startup, the Eclipse GlassFish reads information from the configuration file and automatically adds JNDI database names to the name space, one of which is `jdbc/_default`.

Jakarta EE application clients, enterprise beans, and web components must have access to a JNDI naming environment.

The application component's naming environment is the mechanism that allows customization of

the application component's business logic during deployment or assembly. This environment allows you to customize the application component without needing to access or change the source code off the component. A Jakarta EE container implements the provides the environment to the application component instance as a JNDI naming context.

## How the Naming Environment and the Container Work Together

The application component's environment is used as follows:

- The application component's business methods access the environment using the JNDI interfaces. In the deployment descriptor, the application component provider declares all the environment entries that the application component expects to be provided in its environment at runtime.
- The container provides an implementation of the JNDI naming context that stores the application component environment. The container also provides the tools that allow the deployer to create and manage the environment of each application component.
- A deployer uses the tools provided by the container to initialize the environment entries that are declared in the application component's deployment descriptor. The deployer sets and modifies the values of the environment entries.
- The container makes the JNDI context available to the application component instances at runtime. These instances use the JNDI interfaces to obtain the values of the environment entries.

Each application component defines its own set of environment entries. All instances of an application component within the same container share the same environment entries. Application component instances are not allowed to modify the environment at runtime.

## Naming References and Binding Information

A resource reference is an element in a deployment descriptor that identifies the component's coded name for the resource. For example, `jdbc/SavingsAccountDB`. More specifically, the coded name references a connection factory for the resource.

The JNDI name of a resource and the resource reference name are not the same. This approach to naming requires that you map the two names before deployment, but it also decouples components from resources. Because of this decoupling, if at a later time the component needs to access a different resource, the name does not need to change. This flexibility makes it easier for you to assemble Jakarta EE applications from preexisting components.

The following table lists JNDI lookups and their associated resource references for the Jakarta EE resources used by the Eclipse GlassFish.

Table 18-1 JNDI Lookup Names and Their Associated References

JNDI Lookup Name	Associated Resource Reference
<code>java:comp/env</code>	Application environment entries
<code>java:comp/env/jdbc</code>	JDBC DataSource resource manager connection factories
<code>java:comp/env/ejb</code>	EJB References
<code>java:comp/UserTransaction</code>	UserTransaction references
<code>java:comp/env/mail</code>	JavaMail Session Connection Factories
<code>java:comp/env/url</code>	URL Connection Factories
<code>java:comp/env/jms</code>	JMS Connection Factories and Destinations
<code>java:comp/ORB</code>	ORB instance shared across application components

## Administering JNDI Resources

Within Eclipse GlassFish, you can configure your environment for custom and external JNDI resources. A custom resource accesses a local JNDI repository; an external resource accesses an external JNDI repository. Both types of resources need user-specified factory class elements, JNDI name attributes, and so on.

- [Administering Custom JNDI Resources](#)
- [Administering External JNDI Resources](#)

### Administering Custom JNDI Resources

A custom resource specifies a custom server-wide resource object factory that implements the `javax.naming.spi.ObjectFactory` interface.

The following topics are addressed here:

- [To Create a Custom JNDI Resource](#)
- [To List Custom JNDI Resources](#)
- [To Update a Custom JNDI Resource](#)
- [To Delete a Custom JNDI Resource](#)

#### To Create a Custom JNDI Resource

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

1. Ensure that the server is running. Remote subcommands require a running server.
2. Create a custom resource by using the `create-custom-resource` subcommand.

Information on properties for the subcommand is contained in this help page.

### 3. Restart Eclipse GlassFish.

See [To Restart a Domain](#).

#### Example 18-1 Creating a Custom Resource

This example creates a custom resource named `sample-custom-resource`.

```
asadmin> create-custom-resource --restype topic --factoryclass com.imq.topic  
sample_custom_resource  
Command create-custom-resource executed successfully.
```

#### See Also

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

#### To List Custom JNDI Resources

Use the `list-custom-resources` subcommand in remote mode to list the existing custom resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the custom resources by using the `list-custom-resources` subcommand.

#### Example 18-2 Listing Custom Resources

This example lists the existing custom resources.

```
asadmin> list-custom-resources  
sample_custom_resource01  
sample_custom_resource02  
Command list-custom-resources executed successfully
```

#### See Also

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

#### To Update a Custom JNDI Resource

1. List the custom resources by using the `list-custom-resources` subcommand.
2. Use the `set` subcommand to modify a custom JNDI resource.

### Example 18-3 Updating a Custom JNDI Resource

This example modifies a custom resource.

```
asadmin> set server.resources.custom-resource.custom  
/my-custom-resource.property.value=2010server.resources.custom-resource.custom  
/my-custom-resource.property.value=2010
```

### To Delete a Custom JNDI Resource

Use the `delete-custom-resource` subcommand in remote mode to delete a custom resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the custom resources by using the `list-custom-resources` subcommand.
3. Delete a custom resource by using the `delete-custom-resource` subcommand.

### Example 18-4 Deleting a Custom Resource

This example deletes a custom resource named `sample-custom-resource`.

```
asadmin> delete-custom-resource sample_custom_resource  
Command delete-custom-resource executed successfully.
```

### See Also

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

## Administering External JNDI Resources

Applications running on Eclipse GlassFish often require access to resources stored in an external JNDI repository. For example, generic Java objects might be stored in an LDAP server according to the Java schema. External JNDI resource elements let you configure such external resource repositories.

The following topics are addressed here:

- [To Register an External JNDI Resource](#)
- [To List External JNDI Resources](#)
- [To List External JNDI Entries](#)
- [To Update an External JNDI Resource](#)
- [To Delete an External JNDI Resource](#)

- [Example of Using an External JNDI Resource](#)
- [To Disable Eclipse GlassFish v2 Vendor-Specific JNDI Names](#)

## To Register an External JNDI Resource

Use the `create-jndi-resource` subcommand in remote mode to register an external JNDI resource.

### Before You Begin

The external JNDI factory must implement the `javax.naming.spi.InitialContextFactory` interface.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Register an external JNDI resource by using the `create-jndi-resource` subcommand.

Information on properties for the subcommand is contained in this help page.

3. Restart Eclipse GlassFish.

See [To Restart a Domain](#).

### Example 18-5 Registering an External JNDI Resource

In This example `sample_jndi_resource` is registered.

```
asadmin> create-jndi-resource --jndilookupname sample_jndi  
--restype queue --factoryclass sampleClass --description "this is a sample jndi  
resource" sample_jndi_resource  
Command create-jndi-resource executed successfully
```

### See Also

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

## To List External JNDI Resources

Use the `list-jndi-resources` subcommand in remote mode to list all existing JNDI resources.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the existing JNDI resources by using the `list-jndi-resources` subcommand.

### Example 18-6 Listing JNDI Resources

This example lists the JNDI resources.

```
asadmin> list-jndi-resources
jndi_resource1
jndi_resource2
jndi_resource3
Command list-jndi-resources executed successfully
```

## See Also

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

## To List External JNDI Entries

Use the `list-jndi-entries` subcommand in remote mode to browse and list the entries in the JNDI tree. You can either list all entries, or you can specify the JNDI context or subcontext to list specific entries.

1. Ensure that the server is running. Remote subcommands require a running server.
2. List the JNDI entries for a configuration by using the `list-jndi-entries` subcommand.

### Example 18-7 Listing JNDI Entries

This example lists all the JNDI entries for the naming service.

```
asadmin> list-jndi-entries
jndi_entry03
jndi_entry72
jndi_entry76
Command list-jndi-resources executed successfully
```

## See Also

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

## To Update an External JNDI Resource

1. List the existing JNDI resources by using the `list-jndi-resources` subcommand.
2. Use the `set` subcommand to modify an external JNDI resource.

### Example 18-8 Updating an External JNDI Resource

This example modifies an external resource.

```
asadmin> set server.resources.external-jndi-resource.my-jndi-resource.jndi-lookup-name=bar server.resources.external-jndi-resource.my-jndi-resource.jndi-lookup-name=bar
```

## To Delete an External JNDI Resource

Use the `delete-jndi-resource` subcommand in remote mode to remove a JNDI resource.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Remove an external JNDI entry by using the `delete-jndi-resource` subcommand.

### Example 18-9 Deleting an External JNDI Resource

This example deletes an external JNDI resource:

```
asadmin> delete-jndi-resource jndi_resource2
Command delete-jndi-resource executed successfully.
```

## See Also

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

## Example of Using an External JNDI Resource

```
<resources>
  <!-- external-jndi-resource element specifies how to access Jakarta EE resources
       stored in an external JNDI repository. This example
       illustrates how to access a java object stored in LDAP.
       factory-class element specifies the JNDI InitialContext factory that
       needs to be used to access the resource factory. property element
       corresponds to the environment applicable to the external JNDI context
       and jndi-lookup-name refers to the JNDI name to lookup to fetch the
       designated (in this case the java) object.
  -->
  <external-jndi-resource jndi-name="test/myBean"
    jndi-lookup-name="cn=myBean"
    res-type="test.myBean"
    factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
```

```
</resources>
```

## To Disable Eclipse GlassFish v2 Vendor-Specific JNDI Names

The EJB 3.1 specification supported by Eclipse GlassFish 7 defines portable EJB JNDI names. Because of this, there is less need to continue to use older vendor-specific JNDI names.

By default, Eclipse GlassFish v2-specific JNDI names are applied automatically by Eclipse GlassFish 7 for backward compatibility. However, this can lead to some ease-of-use issues. For example, deploying two different applications containing a Remote EJB component that exposes the same remote interface causes a conflict between the default JNDI names.

The default handling of v2-specific JNDI names in Eclipse GlassFish 7 can be managed with the `asadmin` command or with the `disable-nonportable-jndi-names` boolean property for the `ejb-container` element in `glassfish-ejb-jar.xml`.

Use the `asadmin` command or directly modify the `glassfish-ejb-jar.xml` file to set the `disable-nonportable-jndi-names` property.

- Using the `asadmin` command:

```
asadmin> set server.ejb-container.property.disable-nonportable-jndi-names="true"
```

- Directly modifying the `glassfish-ejb-jar.xml` file.

1. Add the `disable-nonportable-jndi-names` property to the `ejb-container` element in `glassfish-ejb-jar.xml`.

2. Set the value of the `disable-nonportable-jndi-names` boolean, as desired.

`false` - Enables the automatic use of Eclipse GlassFish v2-specific JNDI names. This is the default setting.

`true` - Disables the automatic use of v2-specific JNDI names. In all cases, 5.0-compatible JNDI names will be used.

3. Save the `glassfish-ejb-jar.xml` file and restart the Eclipse GlassFish domain.

This setting applies to all EJBs deployed to the server.

# 19 Administering Transactions

This chapter discusses how to manage the transaction service for the Eclipse GlassFish environment by using the `asadmin` command-line utility. Instructions for manually recovering transactions are also included.

The following topics are addressed here:

- [About Transactions](#)
- [Configuring the Transaction Service](#)
- [Managing the Transaction Service for Rollbacks](#)
- [Recovering Transactions](#)
- [Transaction Logging](#)

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

For more information about the Java Transaction API (JTA) and Java Transaction Service (JTS), see the following sites: <http://www.oracle.com/technetwork/java/javase/tech/jta-138684.html> and <http://www.oracle.com/technetwork/java/javase/tech/jts-140022.html>.

You might also want to read "[Transactions](#)" in The Jakarta EE Tutorial.

## About Transactions

A transaction is a series of discreet actions in an application that must all complete successfully. By enclosing one or more actions in an indivisible unit of work, a transaction ensures data integrity and consistency. If all actions do not complete, the changes are rolled back.

For example, to transfer funds from a checking account to a savings account, the following steps typically occur:

1. Check to see if the checking account has enough money to cover the transfer.
2. Debit the amount from the checking account.
3. Credit the amount to the savings account.
4. Record the transfer to the checking account log.
5. Record the transfer to the savings account log.

These steps together are considered a single transaction.

If all the steps complete successfully, the transaction is committed . If any step fails, all changes from the preceding steps are rolled back, and the checking account and savings account are returned to the states they were in before the transaction started. This type of event is called a rollback. A normal transaction ends in either a committed state or a rolled back state.

The following elements contribute to reliable transaction processing by implementing various APIs and functionalities:

- Transaction Manager. Provides the services and management functions required to support transaction demarcation, transactional resource management, synchronization, and transaction context propagation.
- Eclipse GlassFish. Provides the infrastructure required to support the application runtime environment that includes transaction state management.
- Resource Manager. Through a resource adapter, the resource manager provides the application access to resources. The resource manager participates in distributed transactions by implementing a transaction resource interface used by the transaction manager to communicate transaction association, transaction completion, and recovery work. An example of such a resource manager is a relational database server.
- Resource Adapter. A system-level software library is used by Eclipse GlassFish or a client to connect to a resource manager. A resource adapter is typically specific to a resource manager. The resource adapter is available as a library and is used within the address space of the client using it. An example of such a resource adapter is a Java Database Connectivity (JDBC) driver. For information on supported JDBC drivers, see [Configuration Specifics for JDBC Drivers](#).
- Transactional User Application. In the Eclipse GlassFish environment, the transactional user application uses Java Naming and Directory Interface (JNDI) to look up transactional data sources and, optionally, the user transaction). The application might use declarative transaction attribute settings for enterprise beans, or explicit programmatic transaction demarcation. For more information, see "[The Transaction Manager, the Transaction Synchronization Registry, and UserTransaction](#)" in Eclipse GlassFish Application Development Guide.

The following topics are addressed here:

- [Transaction Resource Managers](#)
- [Transaction Scope](#)

## Transaction Resource Managers

There are three types of transaction resource managers:

- Databases - Use of transactions prevents databases from being left in inconsistent states due to incomplete updates. For information about JDBC transaction isolation levels, see "[Using JDBC Transaction Isolation Levels](#)" in Eclipse GlassFish Application Development Guide.

The Eclipse GlassFish supports a variety of JDBC XA drivers. For a list of the JDBC drivers currently supported by the Eclipse GlassFish, see the [Eclipse GlassFish Release Notes](#). For configurations of supported and other drivers, see [Configuration Specifics for JDBC Drivers](#).

- Java Message Service (JMS) Providers - Use of transactions ensures that messages are reliably delivered. The Eclipse GlassFish is integrated with Open Message Queue, a fully capable JMS provider. For more information about transactions and the JMS API, see [Administering the Java](#)

## [Message Service \(JMS\)](#)

- J2EE Connector Architecture (CA) components - Use of transactions prevents legacy EIS systems from being left in inconsistent states due to incomplete updates. For more information about connectors, see [Administering EIS Connectivity](#).

## Transaction Scope

A local transaction involves only one non-XA resource and requires that all participating application components execute within one process. Local transaction optimization is specific to the resource manager and is transparent to the Jakarta EE application.

In the Eclipse GlassFish, a JDBC resource is non-XA if it meets either of the following criteria:

- In the JDBC connection pool configuration, the DataSource class does not implement the javax.sql.XADataSource interface.
- The Resource Type setting is not set to javax.sql.XADataSource .

A transaction remains local if the following conditions remain true:

- One and only one non-XA resource is used. If any additional non-XA resource is used, the transaction is aborted, because the transaction manager must use XA protocol to commit two or more resources.
- No transaction importing or exporting occurs.

Transactions that involve multiple resources or multiple participant processes are distributed or global transactions. A global transaction can involve one non-XA resource if last agent optimization is enabled. Otherwise, all resources must be XA. The `use-last-agent-optimization` property is set to `true` by default. For details about how to set this property, see [Configuring the Transaction Service](#).

If only one XA resource is used in a transaction, one-phase commit occurs, otherwise the transaction is coordinated with a two-phase commit protocol.

A two-phase commit protocol between the transaction manager and all the resources enlisted for a transaction ensures that either all the resource managers commit the transaction or they all abort. When the application requests the commitment of a transaction, the transaction manager issues a `PREPARE_TO_COMMIT` request to all the resource managers involved. Each of these resources can in turn send a reply indicating whether it is ready for commit (`PREPARED`) or not (`NO`). Only when all the resource managers are ready for a commit does the transaction manager issue a commit request (`COMMIT`) to all the resource managers. Otherwise, the transaction manager issues a rollback request (`ABORT`) and the transaction is rolled back.

## Configuring the Transaction Service

You can configure the transaction service in the Eclipse GlassFish in the following ways:

- To configure the transaction service using the Administration Console, open the Transaction Service component under the relevant configuration. For details, click the Help button in the Administration Console.
- To configure the transaction service, use the `set` subcommand to set the following attributes.

The following examples show the `server-config` configuration, but values for any configuration can be set. For example, if you create a cluster named `cluster1` and a configuration named `cluster1-config` is automatically created for it, you can use `cluster1-config` in the `set` subcommand to get the transaction service settings for that cluster.

```
server-config.transaction-service.automatic-recovery = false
server-config.transaction-service.heuristic-decision = rollback
server-config.transaction-service.keypoint-interval = 2048
server-config.transaction-service.retry-timeout-in-seconds = 600
server-config.transaction-service.timeout-in-seconds = 0
server-config.transaction-service.tx-log-dir = domain-dir/logs
```

You can also set these properties:

```
server-config.transaction-service.property.oracle-xa-recovery-workaround = true
server-config.transaction-service.property.sybase-xa-recovery-workaround = false
server-config.transaction-service.property.disable-distributed-transaction-logging
= false
server-config.transaction-service.property.xaresource-txn-timeout = 0
server-config.transaction-service.property.pending-txn-cleanup-interval = -1
server-config.transaction-service.property.use-last-agent-optimization = true
server-config.transaction-service.property.delegated-recovery = false
server-config.transaction-service.property.wait-time-before-recovery-insec = 60
server-config.transaction-service.property.purge-cancelled-transactions-after = 0
server-config.transaction-service.property.commit-one-phase-during-recovery = false
server-config.transaction-service.property.add-wait-point-during-recovery = 0
server-config.transaction-service.property.db-logging-resource = jdbc/TxnDS
server-config.transaction-service.property.xa-servername = myserver
```

Default property values are shown where they exist. For `db-logging-resource` and `xa-servername`, typical values are shown. Values that are not self-explanatory are as follows:

- The `xaresource-txn-timeout` default of `0` means there is no timeout. The units are seconds.
- The `pending-txn-cleanup-interval` default of `-1` means the periodic recovery thread doesn't run. The units are seconds.
- The `purge-cancelled-transactions-after` default of `0` means cancelled transactions are not purged. The units are the number of cancellations in between purging attempts.
- The `add-wait-point-during-recovery` property does not have a default value. If this property is unset, recovery does not wait. The units are seconds.
- The `db-logging-resource` property does not have a default value. It is unset by default. However, if you set `db-logging-resource` to an empty value, the value used is `jdbc/TxnDS`.

- The `xa-servername` property does not have a default value. Use this property to override server names that can cause errors.

You can use the `get` subcommand to list all the transaction service attributes and the properties that have been set. For details, see the [Eclipse GlassFish Reference Manual](#).

Changing `keypoint-interval`, `retry-timeout-in-seconds`, or `timeout-in-seconds` does not require a server restart. Changing other attributes or properties requires a server restart.

- You can also set the following system properties:

```
ALLOW_MULTIPLE_ENLISTS_DELISTS=false  
JTA_RESOURCE_TABLE_MAX_ENTRIES=8192  
JTA_RESOURCE_TABLE_DEFAULT_LOAD_FACTOR=0.75f
```

The `JTA_RESOURCE_TABLE_DEFAULT_LOAD_FACTOR` default is the default Map resizing value.

## Managing the Transaction Service for Rollbacks

You can roll back a single transaction by using the `asadmin` subcommands described in this section. To do so, the transaction service must be stopped (and later restarted), allowing you to see the active transactions and correctly identify the one that needs to be rolled back.

The following topics are addressed here:

- [To Stop the Transaction Service](#)
- [To Roll Back a Transaction](#)
- [To Restart the Transaction Service](#)
- [Determining Local Transaction Completion at Shutdown](#)

### To Stop the Transaction Service

Use the `freeze-transaction-service` subcommand in remote mode to stop the transaction service. When the transaction service is stopped, all in-flight transactions are immediately suspended. You must stop the transaction service before rolling back any in-flight transactions.

Running this subcommand on a stopped transaction subsystem has no effect. The transaction service remains suspended until you restart it by using the `unfreeze-transaction-service` subcommand.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Stop the transaction service by using the `freeze-transaction-service` subcommand.

#### Example 19-1 Stopping the Transaction Service

This example stops the transaction service.

```
asadmin> freeze-transaction-service --target instance1  
Command freeze-transaction-service executed successfully
```

## See Also

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

## To Roll Back a Transaction

In some situations, you might want to roll back a particular transaction. Before you can roll back a transaction, you must first stop the transaction service so that transaction operations are suspended. Use the `rollback-transaction` subcommand in remote mode to roll back a specific transaction.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Enable monitoring using the `set` subcommand. For example:

```
asadmin> set cluster1-config.monitoring-service.module-monitoring-  
levels.transaction-service=HIGH
```

3. Use the `freeze-transaction-service` subcommand to halt in-process transactions. See [To Stop the Transaction Service](#).
4. Identify the ID of the transaction you want to roll back.

To see a list of IDs of active transactions, use the `get` subcommand with the `--monitor` option to get the monitoring data for the `activeids` statistic. See [Transaction Service Statistics](#). For example:

```
asadmin> get --monitor instance1.server.transaction-service.activeids-current
```

5. Roll back the transaction by using the `rollback-transaction` subcommand.

The transaction is not rolled back at the time of this command's execution, but only marked for rollback. The transaction is rolled back when it is completed.

### Example 19-2 Rolling Back a Transaction

This example rolls back the transaction with transaction ID `00000000000001_00`.

```
asadmin> rollback-transaction --target instance1 00000000000001_00
```

```
Command rollback-transaction executed successfully
```

## See Also

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

## To Restart the Transaction Service

Use the `unfreeze-transaction-service` subcommand in remote mode to resume all the suspended in-flight transactions. Run this subcommand to restart the transaction service after it has been frozen.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Restart the suspended transaction service by using the `unfreeze-transaction-service` subcommand.

### Example 19-3 Restarting the Transaction Service

This example restarts the transaction service after it has been frozen.

```
asadmin> unfreeze-transaction-service --target instance1
Command unfreeze-transaction-service executed successfully
```

## See Also

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

## Determining Local Transaction Completion at Shutdown

When you shut down a Eclipse GlassFish instance, all database connections are closed. When an Oracle JDBC driver-based database connection is closed in the middle of a non-XA transaction, all pending changes are committed. Other databases usually roll back pending changes when a connection is closed without being explicitly committed. To determine the exact behavior for your database, refer to the documentation from your JDBC driver vendor.

To explicitly specify whether Eclipse GlassFish commits or rolls back non-XA transactions at server shutdown, set the `com.sun.enterprise.in-progress-local-transaction.completion-mode` JVM option to either `commit` or `rollback` using the `create-jvm-options` subcommand. For example:

```
asadmin> create-jvm-options -Dcom.sun.enterprise.in-progress-local
-tx-completion-mode=rollback
```

# Recovering Transactions

There are some situations where the commit or rollback operations might be interrupted, typically because the server crashed or a resource manager crashed. Crash situations can leave some transactions stranded between steps. Eclipse GlassFish is designed to recover from these failures. If the failed transaction spans multiple servers, the server that started the transaction can contact the other servers to get the outcome of the transaction. If the other servers are unreachable, the transaction uses heuristic decision information to determine the outcome.

The following topics are addressed here:

- [Automatic Transaction Recovery](#)
- [To Manually Recover Transactions](#)
- [Distributed Transaction Recovery](#)
- [Recovery Workarounds and Limitations](#)

## Automatic Transaction Recovery

Eclipse GlassFish can perform automatic recovery in these ways:

- Pending transactions are completed upon server startup if `automatic-recovery` is set to `true`.
- Periodic automatic recovery is performed by a background thread if the `pending-txn-cleanup-interval` property is set to a positive value.

Changing these settings requires a server restart. For more information about how to change these settings, see [Configuring the Transaction Service](#).

If commit fails during recovery, a message is written to the server log.

## To Manually Recover Transactions

Use the `recover-transactions` subcommand in remote mode to manually recover transactions that were pending when a resource or a server instance failed.

For a standalone server, do not use manual transaction recovery to recover transactions after a server failure. For a standalone server, manual transaction recovery can recover transactions only when a resource fails, but the server is still running. If a standalone server fails, only the full startup recovery process can recover transactions that were pending when the server failed.

For an installation of multiple server instances, you can use manual transaction recovery from a surviving server instance to recover transactions after a server failure. For manual transaction recovery to work properly, transaction logs must be stored on a shared file system that is accessible

to all server instances. See [Transaction Logging](#).

When you execute `recover-transactions` in non-delegated mode, you can recover transactions that didn't complete two-phase commit because of a resource crash. To use manual transaction recovery in this way, the following conditions must be met:

- The `recover-transactions` command should be executed after the resource is restarted.
- Connection validation should be enabled so the connection pool is refreshed when the resource is accessed after the recovery. For more information, see "[Connection Validation Settings](#)" in Eclipse GlassFish Performance Tuning Guide.

If commit fails during recovery, a message is written to the server log.

A JMS resource crash is handled the same way as any other resource.



You can list in-doubt Open Message Queue transactions using the `imqcmd list txn` subcommand. For more information, see "[Managing Transactions](#)" in Open Message Queue Administration Guide.

1. Ensure that the server is running. Remote subcommands require a running server.
2. Manually recover transactions by using the `recover-transactions` subcommand.

#### Example 19-4 Manually Recovering Transactions

This example performs manual recovery of transactions on `instance1`, saving them to `instance2`.

```
asadmin recover-transactions --target instance2 instance1
Transaction recovered.
```

#### See Also

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

## Distributed Transaction Recovery

To enable cluster-wide automatic recovery, you must first facilitate storing of transaction logs in a shared file system. See [Transaction Logging](#).

Next, you must set the transaction service's `delegated-recovery` property to `true` (the default is `false`). For information about setting `tx-log-dir` and `delegated-recovery`, see [Configuring the Transaction Service](#).

## Recovery Workarounds and Limitations

The Eclipse GlassFish provides workarounds for some known issues with transaction recovery implementations.



These workarounds do not imply support for any particular JDBC driver.

### General Recovery Limitations

The following general limitations apply to transaction recovery:

- Recovery succeeds if there are no exceptions during the process. This is independent of the number of transactions that need to be recovered.
- Only transactions that did not complete the two-phase commit can be recovered (one of the XA resources failed or Eclipse GlassFish crashed after resources were prepared).
- Manual transaction recovery cannot recover transactions after a server crash on a standalone server instance. Manual operations are intended for cases when a resource dies unexpectedly while the server is running. In case of a server crash, only startup recovery can recover in-doubt transactions.
- It is not possible to list transaction IDs for in-doubt transactions.
- Delegated transaction recovery (by a different server instance in a cluster) is not possible if the failed instance used an **EMBEDDED** Message Queue broker, or if it used a **LOCAL** or **REMOTE** Message Queue broker and the broker also failed. In this case, only automatic recovery on server instance restart is possible. This is because for conventional Message Queue clustering, state information in a failed broker is not available until the broker restarts.

### Oracle Setup for Transaction Recovery

You must configure the following **grant** statements in your Oracle database to set up transaction recovery:

```
grant select on SYS.DBA_PENDING_TRANSACTIONS to user;
grant execute on SYS.DBMS_SYSTEM to user;
grant select on SYS.PENDING_TRANS$ to user;
grant select on SYS.DBA_2PC_NEIGHBORS to user;
grant execute on SYS.DBMS_XA to user;
grant select on SYS.DBA_2PC_PENDING to user;
```

The user is the database administrator. On some versions of the Oracle driver the last **grant execute** fails. You can ignore this.

## Oracle Thin Driver

In the Oracle thin driver, the `XAResource.recover` method repeatedly returns the same set of in-doubt Xids regardless of the input flag. According to the XA specifications, the Transaction Manager initially calls this method with TMSTARTSCAN and then with TMNOFLAGS repeatedly until no Xids are returned. The `XAResource.commit` method also has some issues.

To disable the Eclipse GlassFish workaround, set the `oracle-xa-recovery-workaround` property value to `false`. For details about how to set this property, see [Configuring the Transaction Service](#). This workaround is used unless explicitly disabled.

## Delegated Recovery After Server Crash Doesn't Work on MySQL

The MySQL database supports XA transaction recovery only when the database crashes. When a Eclipse GlassFish instance crashes, MySQL rolls back prepared transactions.

## Call to `XATeminator.recover()` During `ResourceAdapter.start()` Hangs If Automatic Recovery Is Enabled

Calls to `XATeminator.recover()` from the `ResourceAdapter.start()` method never return because Eclipse GlassFish deadlocks. This only occurs when automatic recovery is enabled.

It is not advisable to do transactional activities, such as starting a transaction or calling `XATeminator.recover()`, during `ResourceAdapter.start()`. For more information, see <http://markmail.org/message/ogc7qndhaywfkdrp#query:+page:1+mid:kyyzpcexusbnv7ri+state:results>.

## Transaction Logging

The transaction service writes transactional activity into transaction logs so that transactions can be recovered. You can control transaction logging in these ways:

- Set the location of the transaction log files in one of these ways:
  - Set the Eclipse GlassFish's `log-root` setting to a shared file system base directory and set the transaction service's `tx-log-dir` attribute to a relative path.
  - Set `tx-log-dir` to an absolute path to a shared file system directory, in which case `log-root` is ignored for transaction logs.
  - Set a system property called `TX-LOG-DIR` to a shared file system directory. For example:

```
asadmin> create-system-properties --target server TX-LOG-DIR=/inst1/logs
```

For information about setting `log-root` and other general logging settings, see [Administering the Logging Service](#).

- Turn off transaction logging by setting the `disable-distributed-transaction-logging` property to `true` and the `automatic-recovery` attribute to `false`. Do this only if performance is more important than transaction recovery.

All instances in a cluster must be owned by the same user (`uid`), and read/write permissions for that user must be set on the transaction log directories.



Transaction logs should be stored in a high-availability network file system (NFS) to avoid a single point of failure.

## To Store Transaction Logs in a Database

For multi-core machines, logging transactions to a database may be more efficient. Transaction logging is designed to work with any JDBC-compliant database. For databases with which transaction logging has been tested, see the Eclipse GlassFish Release Notes.

- Create a JDBC connection Pool. To use non-transactional connections to insert log records, you can either set the `non-transactional-connections` attribute to `true` in this step, or you can perform step 5 later.
- Create a JDBC resource that uses the connection pool and note the JNDI name of the JDBC resource.
- Automatic table creation for the transaction logs is done by default. However, if you would prefer to create the table manually, name it `txn_log_table` with the following schema:

Column Name	JDBC Type
LOCALTID	VARCHAR
INSTANCENAME	VARCHAR
SERVERTNAME	VARCHAR(n)
GTRID	VARBINARY

The size of the `SERVERTNAME` column should be at least the length of the Eclipse GlassFish host name plus 10 characters.

The size of the `GTRID` column should be at least 64 bytes.

- Add the `db-logging-resource` property to the transaction service. For example:

```
asadmin set server-config.transaction-service.property.db-logging-
resource="jdbc/TxnDS"
```

The property's value should be the JNDI name of the JDBC resource configured previously.

- If you didn't set the `non-transactional-connections` attribute to `true` in step 1 and you want to use non-transactional connections to insert log records, use the following `asadmin create-jvm-options` command to reference an existing transactional resource but use non-transactional

connections for the **INSERT** statements:

```
asadmin create-jvm-options  
-Dcom.sun.jts.dblogging.use.nontx.connection.for.add=true
```

6. To disable file synchronization, use the following **asadmin create-jvm-options** command:

```
asadmin create-jvm-options -Dcom.sun.appserv.transaction.nofdsync
```

7. Restart the server.

## Next Steps

To define the SQL used by the transaction manager when it is storing its transaction logs in the database, use the following flags:

- Dcom.sun.jts.dblogging.insertquery=**sql** statement
- Dcom.sun.jts.dblogging.deletequery=**sql** statement
- Dcom.sun.jts.dblogging.selectquery=**sql** statement
- Dcom.sun.jts.dblogging.selectservernamequery=**sql** statement

The default statements are as follows:

- Dcom.sun.jts.dblogging.insertquery=insert into txn\_log\_table values ( ?, ?, ?, ?, ? )
- Dcom.sun.jts.dblogging.deletequery=delete from txn\_log\_table where localtid = ? and servername = ?
- Dcom.sun.jts.dblogging.selectquery=select \* from txn\_log\_table where servername = ?
- Dcom.sun.jts.dblogging.selectservernamequery=select distinct servername from txn\_log\_table where instancename = ?

To set one of these flags using the **asadmin create-jvm-options** command, you must quote the statement. For example:

```
create-jvm-options '-Dcom.sun.jts.dblogging.deletequery=delete from txn_log_table where gtrid = ?'
```

You can also set JVM options in the Administration Console. Select the JVM Settings component under the relevant configuration. These flags and their statements must also be quoted in the Administration Console. For example:

```
'-Dcom.sun.jts.dblogging.deletequery=delete from txn_log_table where gtrid = ?'
```

## See Also

For information about JDBC connection pools and resources, see [Administering Database Connectivity](#). For more information about the **asadmin set** and **asadmin create-jvm-options**

commands, see the [Eclipse GlassFish Reference Manual](#). For databases with which transaction logging has been tested, see the Eclipse GlassFish Release Notes.

## **Part III**

# Appendices

# A Subcommands for the `asadmin` Utility

This appendix lists the `asadmin` subcommands that are included with this release of the Eclipse GlassFish 7 software.

- [General Administration Subcommands](#)
- [Batch Jobs Subcommands](#)
- [Concurrent Resources Subcommands](#)
- [Connectivity Subcommands](#)
- [Domain Subcommands](#)
- [Internet Connectivity Subcommands](#)
- [JavaMail Subcommands](#)
- [JMS Subcommands](#)
- [JNDI Subcommands](#)
- [JVM Subcommands](#)
- [Life Cycle Module Subcommands](#)
- [Logging and Monitoring Subcommands](#)
- [ORB Subcommands](#)
- [Thread Pool Subcommands](#)
- [Transaction Service Subcommands](#)

For information and instructions on using the `asadmin` application deployment subcommands, see the [Eclipse GlassFish Application Deployment Guide](#).

Online help for the `asadmin` subcommands can be invoked on the command line, for example, `asadmin create-domain help`. The [Eclipse GlassFish Reference Manual](#) also provides a collection of these help pages.



The common options used with remote subcommands are described in the [oasadmin\(1M\)](#) help page.

## General Administration Subcommands

### `add-resources`

Creates the resources named in the specified XML file. Supported in remote mode only. For procedural information in this guide, see [To Add Resources From an XML File](#).

### `asadmin`

Describes how the `asadmin` utility works.

## **attach**

Attaches to subcommands that were started using the `asadmin --detach` option or that contain progress information. For procedural information in this guide, see [To Run asadmin Subcommands in --detach Mode](#).

## **configure-managed-jobs**

Configures how long information about subcommands (jobs) that were started using the `asadmin --detach` option or that contain progress information is kept. For procedural information in this guide, see [To Run asadmin Subcommands in --detach Mode](#).

## **create-module-config**

Adds the default configuration of a module to `domain.xml`.

## **create-service**

Configures the starting of a domain administration server (DAS) on an unattended boot. On Oracle Solaris 10, this subcommand uses the Service Management Facility (SMF). For procedural information in this guide, see [To Configure a DAS or an Instance for Automatic Restart on Oracle Solaris](#).

## **create-system-properties**

Creates or updates system properties. Supported in remote mode only. For procedural information in this guide, see [To Create System Properties](#).

## **delete-module-config**

Removes the configuration of a module from `domain.xml`.

## **delete-system-property**

Deletes system properties of a domain or configuration, or server instance. Supported in remote mode only. For procedural information in this guide, see [To Delete a System Property](#).

## **get**

Gets an attribute of an element in the `domain.xml` file. With the `-m` option, gets the names and values of the monitorable or configurable attributes. For procedural information in this guide, see [Guidelines for Using the list and get Subcommands for Monitoring](#).

## **get-active-module-config**

Displays the current active configuration of a module.

## **list**

Lists the configurable element. On Oracle Solaris, quotes are needed when running subcommands with `*` as the option value or operand. For procedural information in this guide, see [Guidelines for Using the list and get Subcommands for Monitoring](#).

## **list-commands**

Lists all the `asadmin` subcommands, local subcommands first, then remote subcommands. You can specify that only remote subcommands or only local subcommands be displayed. Supported in remote mode only. For procedural information in this guide, see [To List Subcommands](#).

## **list-containers**

Lists application containers and the status of each container. Supported in remote mode only. For procedural information in this guide, see [To List Containers](#).

## **list-jobs**

Lists information about subcommands that were started using the `asadmin --detach` option or that contain progress information. For procedural information in this guide, see [To Run asadmin Subcommands in --detach Mode](#).

## **list-modules**

Lists modules that are accessible to the Eclipse GlassFish subsystem. The status of each module is included. Supported in remote mode only. For procedural information in this guide, see [To List Modules](#).

## **list-system-properties**

Lists the system properties of a domain or configuration. Supported in remote mode only. For procedural information in this guide, see [To List System Properties](#).

## **list-timers**

List the timers owned by a specific server instance. Supported in remote mode only. For procedural information in this guide, see [To List Timers](#).

## **multimode**

Provides an `asadmin>` prompt for running multiple subcommands while preserving options and environment settings. Supported in local mode only. For procedural information, see [Using the asadmin Utility](#).

## **osgi**

Delegates the command line to the Apache Felix Gogo remote shell for the execution of OSGi shell commands. For procedural information in this guide, see [To Run Apache Felix Gogo Remote Shell Commands](#).

## **osgi-shell**

Provides interactive access to the Apache Felix Gogo remote shell for the execution of OSGi shell commands. For procedural information in this guide, see [To Run Apache Felix Gogo Remote Shell Commands](#).

## **set**

Sets the values of one or more configurable attributes. For procedural information in this guide, see [Configuring Monitoring](#).

## **setup-local-dcom**

Sets up the Distributed Component Object Model (DCOM) remote protocol on the host where the subcommand is run. The DCOM remote protocol is available only on Windows systems.

## **show-component-status**

Lists the status of existing components. Supported in remote mode only. For procedural information in this guide, see [To Show Component Status](#).

### **start-database**

Starts the Apache Derby database server. Use this subcommand only for working with applications deployed to the Eclipse GlassFish. For procedural information in this guide, see [To Start the Database](#).

### **stop-database**

Stops a process of the Apache Derby DB database server. For procedural information in this guide, see [To Stop the Database](#).

### **version**

Displays the version information for the option specified in archive or folder format. Supported in remote mode only. For procedural information in this guide, see [To Display the Eclipse GlassFish Version](#).

## **Batch Jobs Subcommands**

### **list-batch-jobs**

Lists batch jobs and job details. For procedural information in this guide, see [To List Batch Jobs](#).

### **list-batch-job-executions**

Lists batch job executions and execution details. For procedural information in this guide, see [To List Batch Job Executions](#).

### **list-batch-job-steps**

Lists steps for a specific batch job execution. For procedural information in this guide, see [To List Batch Job Steps](#).

### **list-batch-runtime-configuration**

Displays the configuration of the batch runtime. For procedural information in this guide, see [To List the Batch Runtime Configuration](#).

### **set-batch-runtime-configuration**

Configures the batch runtime. For procedural information in this guide, see [To Configure the Batch Runtime](#).

## **Concurrent Resources Subcommands**

### **create-context-service**

Creates a context service resource. For procedural information in this guide, see [To Create a Context Service](#).

### **create-managed-executor-service**

Creates a managed executor service resource. For procedural information in this guide, see [To](#)

Create a Managed Executor Service.

#### **create-managed-scheduled-executor-service**

Creates a managed scheduled executor service resource. For procedural information in this guide, see [To Create a Managed Scheduled Executor Service](#).

#### **create-managed-thread-factory**

Creates a managed thread factory resource. For procedural information in this guide, see [To Create a Managed Thread Factory](#).

#### **list-context-services**

Lists context service resources. For procedural information in this guide, see [To List Context Services](#).

#### **list-managed-executor-services**

Lists managed executor service resources. For procedural information in this guide, see [To List Managed Executor Services](#).

#### **list-managed-scheduled-executor-services**

Lists managed scheduled executor service resources. For procedural information in this guide, see [To List Managed Scheduled Executor Services](#).

#### **list-managed-thread-factories**

Lists managed thread factory resources. For procedural information in this guide, see [To List Managed Thread Factories](#).

#### **delete-context-service**

Removes a context service resource. For procedural information in this guide, see [To Delete a Context Service](#).

#### **delete-managed-executor-service**

Removes a managed executor service resource. For procedural information in this guide, see [To Delete a Managed Executor Service](#).

#### **delete-managed-scheduled-executor-service**

Removes a managed scheduled executor service resource. For procedural information in this guide, see [To Delete a Managed Scheduled Executor Service](#).

#### **delete-managed-thread-factory**

Removes a managed thread factory resource. For procedural information in this guide, see [To Delete a Managed Thread Factory](#).

## **Connectivity Subcommands**

#### **create-admin-object**

Creates an administered object. For procedural information in this guide, see [To Create an](#)

[Administered Object](#).

#### **create-connector-connection-pool**

Adds a new connector connection pool with the specified connection pool name. For procedural information in this guide, see [To Create a Connector Connection Pool](#).

#### **create-connector-resource**

Creates a connector resource. For procedural information in this guide, see [To Create a Connector Resource](#).

#### **create-connector-security-map**

Creates a connector security map for the specified connector connection pool. For procedural information, see [To Create a Connector Security Map](#).

#### **create-connector-work-security-map**

Creates a connector work security map for the specified resource adapter. Supported in remote mode only. For procedural information in this guide, see [To Create a Connector Work Security Map](#).

#### **create-jdbc-resource**

Creates a new JDBC resource. Supported in remote mode only. For procedural information in this guide, see [To Create a JDBC Resource](#).

#### **create-jdbc-connection-pool**

Registers a new JDBC connection pool with the specified JDBC connection pool name. Supported in remote mode only. For procedural information in this guide, see [To Create a JDBC Connection Pool](#).

#### **create-resource-adapter-config**

Creates configuration information for the connector module. Supported in remote mode only. For procedural information in this guide, see [To Create Configuration Information for a Resource Adapter](#).

#### **delete-admin-object**

Deletes an administered object. For procedural information in this guide, see [To Delete an Administered Object](#).

#### **delete-connector-connection-pool**

Removes the connector connection pool specified using the `connector_connection_pool_name` operand. For procedural information in this guide, see [To Delete a Connector Connection Pool](#).

#### **delete-connector-resource**

Deletes connector resource. For procedural information in this guide, see [To Delete a Connector Resource](#).

#### **delete-connector-security-map**

Deletes a specified connector security map. Supported in remote mode only. For procedural information in this guide, see [To Delete a Connector Security Map](#).

## **delete-connector-work-security-map**

Deletes a specified connector work security map. Supported in remote mode only. For procedural information in this guide, see [To Delete a Connector Work Security Map](#).

## **delete-jdbc-connection-pool**

Deletes the specified JDBC connection pool. Supported in remote mode only. For procedural information in this guide, see [To Delete a JDBC Connection Pool](#).

## **delete-jdbc-resource**

Deletes a JDBC resource. The specified JNDI name identifies the resource to be deleted. Supported in remote mode only. For procedural information in this guide, see [To Delete a JDBC Resource](#).

## **delete-resource-adapter-config**

Deletes configuration information for the connector module. Supported in remote mode only. For procedural information in this guide, see [To Delete a Resource Adapter Configuration](#).

## **flush-connection-pool**

Reinitializes all connections established in the specified connection. For procedural information in this guide, see [To Reset \(Flush\) a Connection Pool](#).

## **list-admin-objects**

Lists administered objects. For procedural information in this guide, see [To List Administered Objects](#).

## **list-connector-connection-pools**

Lists the connector connection pools that have been created. For procedural information in this guide, see [To List Connector Connection Pools](#).

## **list-connector-resources**

Creates connector resources. For procedural information in this guide, see [To List Connector Resources](#).

## **list-connector-security-maps**

Lists the connector security maps belonging to a specified connector connection pool. For procedural information in this guide, see [To List Connector Security Maps](#).

## **list-connector-work-security-maps**

Lists the existing connector work security maps for a resource adapter. Supported in remote mode only. For procedural information in this guide, see [To List Connector Work Security Maps](#).

## **list-jdbc-connection-pools**

Lists the existing JDBC connection pools. Supported in remote mode only. For procedural information in this guide, see [To List JDBC Connection Pools](#).

## **list-jdbc-resources**

Lists the existing JDBC resources. Supported in remote mode only. For procedural information in this guide, see [To List JDBC Resources](#).

## **list-resource-adapter-configs**

Lists configuration information for the connector modules. Supported in remote mode only. For procedural information in this guide, see [To List Resource Adapter Configurations](#).

## **ping-connection-pool**

Tests if a JDBC connection pool is usable. Supported in remote mode only. For procedural information in this guide, see [To Contact \(Ping\) a Connection Pool](#).

## **update-connector-security-map**

Modifies a security map for the specified connector connection pool. For procedural information in this guide, see [To Update a Connector Security Map](#).

## **update-connector-work-security-map**

Modifies a work security map that belongs to a specific resource adapter (connector module). For procedure information in this guide, see [To Update a Connector Work Security Map](#).

# Domain Subcommands

## **backup-domain**

Describes how to back up a domain. Supported in local mode only. For procedural information in this guide, see [To Back Up a Domain](#).

## **create-domain**

Creates the configuration of a domain. A domain can exist independent of other domains. Any user who has access to the `asadmin` utility on a given host can create a domain and store its configuration in a location of choice. For procedural information in this guide, see [To Create a Domain](#).

## **delete-domain**

Deletes the specified domain. The domain must be stopped before it can be deleted. For procedural information in this guide, see [To Delete a Domain](#).

## **list-backups**

Lists the existing domain backups. Supported in local mode only. For procedural information in this guide, see [To List Domain Backups](#).

## **list-domains**

Lists the existing domains and their statuses. If the domain directory is not specified, the domains in the domain-root-dir, the default for which is `as-install/domains`, are displayed. For procedural information in this guide, see [To List Domains](#).

## **login**

Allows you to log in to a domain. For procedural information in this guide, see [To Log In to a Domain](#).

### **restart-domain**

Restarts the Domain Administration Server (DAS) of the specified domain. Supported in remote mode only. For procedural information in this guide, see [To Restart a Domain](#).

### **restore-domain**

Recovers and domain from a backup file. Supported in local mode only. For procedural information in this guide, see [To Restore a Domain](#).

### **start-domain**

Starts a domain. If the domain directory is not specified, the default `domain1` in the default domain-root-dir directory is started. If there are two or more domains, the `domain_name` operand must be specified. For procedural information in this guide, see [To Start a Domain](#).

### **stop-domain**

Stops the domain administration server (DAS) of the specified domain. Supported in remote mode only. For procedural information in this guide, see [To Stop a Domain](#).

### **uptime**

Displays the length of time that the domain administration server (DAS) has been running since the last restart. Supported in remote mode only. For procedural information in this guide, see [To Display Domain Uptime](#).

## **Internet Connectivity Subcommands**

### **create-http**

Creates a set of HTTP parameters for a protocol, which in turn configures one or more network listeners. Supported in remote mode only. For procedural information in this guide, see [To Create an HTTP Configuration](#).

### **create-http-listener**

Creates a new HTTP listener socket. Supported in remote mode only. For procedural information in this guide, see [To Create an Internet Connection](#).

### **create-network-listener**

Creates a new HTTP listener socket. Supported in remote mode only. For procedural information in this guide, see [To Create an Internet Connection](#).

### **create-protocol**

Creates a protocol for a listener. Supported in remote mode only. For procedural information in this guide, see [To Create a Protocol](#).

### **create-transport**

Creates a transport for a listener. Supported in remote mode only. For procedural information in this guide, see [To Create a Transport](#).

## **create-virtual-server**

Creates the specified virtual server element. Supported in remote mode only. For procedural information in this guide, see [To Create a Virtual Server](#).

## **create-ssl**

Creates and configures the SSL element in the selected HTTP listener to enable secure communication on that listener/service. Supported in remote mode only. For procedural information in this guide, see [To Configure an HTTP Listener for SSL](#).

## **delete-http**

Deletes an existing HTTP configuration. Supported in remote mode only. For procedural information in this guide, see [To Delete an HTTP Configuration](#).

## **delete-http-listener**

Deletes the specified HTTP listener. Supported in remote mode only. For procedural information in this guide, see [To Delete an HTTP Network Listener](#).

## **delete-network-listener**

Deletes the specified HTTP listener. Supported in remote mode only. For procedural information in this guide, see [To Delete an HTTP Network Listener](#).

## **delete-protocol**

Deletes and existing HTTP protocol. Supported in remote mode only. For procedural information in this guide, see [To Delete a Protocol](#).

## **delete-ssl**

Deletes the SSL element in the selected HTTP listener. Supported in remote mode only. For procedural information in this guide, see [To Delete SSL From an HTTP Listener](#).

## **delete-transport**

Deletes and existing HTTP transport. Supported in remote mode only. For procedural information in this guide, see [To Delete a Transport](#).

## **delete-virtual-server**

Deletes the specified virtual server element. Supported in remote mode only. For procedural information in this guide, see [To Delete a Virtual Server](#).

## **list-http-listeners**

Lists the existing HTTP listeners. Supported in remote mode only. For procedural information in this guide, see [To List HTTP Network Listeners](#).

## **list-network-listeners**

Lists the existing HTTP listeners. Supported in remote mode only. For procedural information in this guide, see [To List HTTP Network Listeners](#).

## **list-protocols**

Lists the existing HTTP protocols. Supported in remote mode only. For procedural information in this guide, see [To List Protocols](#).

## **list-transports**

Lists the existing HTTP transports. Supported in remote mode only. For procedural information in this guide, see [To List Transports](#).

## **list-virtual-servers**

Lists the existing virtual servers. Supported in remote mode only. For procedural information in this guide, see [To List Virtual Servers](#).

# **JavaMail Subcommands**

## **create-javamail-resource**

Creates a JavaMail session resource. Supported in remote mode only. For procedural information in this guide, see [To Create a JavaMail Resource](#).

## **delete-javamail-resource**

Deletes a JavaMail session resource. Supported in remote mode only. For procedural information in this guide, see [To Delete a JavaMail Resource](#).

## **list-javamail-resources**

Creates JavaMail session resources. Supported in remote mode only. For procedural information in this guide, see [To List JavaMail Resources](#).

# **JMS Subcommands**

## **create-jmsdest**

Creates a JMS physical destination. Along with the physical destination, you use the `create-jms-resource` subcommand to create a JMS destination resource that has a `Name` property that specifies the physical destination. Supported in remote mode only. For procedural information in this guide, see [To Create a JMS Physical Destination](#).

## **create-jms-host**

Creates a JMS host within the JMS service. Supported in remote mode only. For procedural information in this guide, see [To Create a JMS Host](#).

## **create-jms-resource**

Creates a JMS connection factory resource or JMS destination resource. Supported in remote mode only. Supported in remote mode only. For procedural information in this guide, see [To Create a Connection Factory or Destination Resource](#).

## **delete-jmsdest**

Removes the specified JMS destination. Supported in remote mode only. For procedural information in this guide, see [To Delete a JMS Physical Destination](#).

### **delete-jms-host**

Deletes a JMS host within the JMS service. Supported in remote mode only. For procedural information in this guide, see [To Delete a JMS Host](#).

### **delete-jms-resource**

Deletes a JMS connection factory resource or JMS destination resource. Supported in remote mode only. For procedural information in this guide, see [To Delete a Connection Factory or Destination Resource](#).

### **flush-jmsdest**

Purges the messages from a physical destination in the specified JMS Service configuration of the specified target. Supported in remote mode only. For procedural information in this guide, see [To Purge Messages From a Physical Destination](#).

### **jms-ping**

Checks if the JMS service (also known as the JMS provider) is up and running. Supported in remote mode only. For procedural information in this guide, see [Troubleshooting the JMS Service](#).

### **list-jmsdest**

Lists the JMS physical destinations. Supported in remote mode only. For procedural information in this guide, see [To List JMS Physical Destinations](#).

### **list-jms-hosts**

Lists the existing JMS hosts. Supported in remote mode only. For procedural information in this guide, see [To List JMS Hosts](#).

### **list-jms-resources**

Lists the existing JMS connection factory or destination resources. Supported in remote mode only. For procedural information in this guide, see [To List JMS Resources](#).

## **JNDI Subcommands**

### **create-custom-resource**

Creates a custom JNDI resource. Supported in remote mode only. For procedural information in this guide, see [To Create a Custom JNDI Resource](#).

### **create-jndi-resource**

Creates an external JNDI resource. Supported in remote mode only. For procedural information in this guide, see [To Register an External JNDI Resource](#).

### **delete-custom-resource**

Deletes a custom JNDI resource. Supported in remote mode only. For procedural information in this guide, see [To Delete a Custom JNDI Resource](#).

## **delete-jndi-resource**

Deletes an external JNDI resource. Supported in remote mode only. For procedural information in this guide, see [To Delete an External JNDI Resource](#).

## **list-custom-resources**

Lists the existing custom JNDI resources. Supported in remote mode only. For procedural information in this guide, see [To List Custom JNDI Resources](#).

## **list-jndi-entries**

Lists the entries in the JNDI tree. Supported in remote mode only. For procedural information in this guide, see [To List External JNDI Entries](#),

## **list-jndi-resources**

Lists the existing external JNDI resources. Supported in remote mode only. For procedural information in this guide, see [To List External JNDI Resources](#).

# JVM Subcommands

## **create-jvm-options**

Creates a JVM option in the Java configuration or profiler elements of the `domain.xml` file. Supported in remote mode only. For procedural information in this guide, see [To Create JVM Options](#).

## **create-profiler**

Creates a profiler element. Supported in remote mode only. For procedural information in this guide, see [To Create a Profiler](#).

## **delete-jvm-options**

Deletes the specified JVM option from the Java configuration or profiler elements of the `domain.xml` file. Supported in remote mode only. For procedural information in this guide, see [To Delete JVM Options](#).

## **delete-profiler**

Deletes the specified profiler element. Supported in remote mode only. For procedural information in this guide, see [To Delete a Profiler](#).

## **generate-jvm-report**

Generates a report showing the threads, classes, and memory for the virtual machine that runs Eclipse GlassFish. For procedural information in this guide, see [To Generate a JVM Report](#).

## **list-jvm-options**

Lists the command-line options that are passed to the Java application launcher when Eclipse GlassFish is started. Supported in remote mode only. For procedural information in this guide, see [To List JVM Options](#).

# Life Cycle Module Subcommands

## **create-lifecycle-module**

Creates a new life cycle module. Supported in remote mode only. For procedural information in this guide, see [To Create a Life Cycle Module](#).

## **list-lifecycle-modules**

Lists life cycle modules. Supported in remote mode only. For procedural information in this guide, see [To List Life Cycle Modules](#).

## **delete-lifecycle-module**

Deletes an existing life cycle module. Supported in remote mode only. For procedural information in this guide, see [To Delete a Life Cycle Module](#).

# Logging and Monitoring Subcommands

## **collect-log-files**

Collects all available log files and creates a ZIP archive. Supported in remote mode only. For procedural information in this guide, see [To Collect Log Files into a ZIP Archive](#).

## **disable-monitoring**

Disables the monitoring service. Supported in remote mode only. For procedural information in this guide, see [To Disable Monitoring](#).

## **enable-monitoring**

Enables the monitoring service. Supported in remote mode only. For procedural information in this guide, see [To Enable Monitoring](#).

## **list-log-attributes**

Lists log file attributes. Supported in remote mode only. For procedural information in this guide, see [Configuring the Logging Service](#).

## **list-log-levels**

Lists the existing loggers. Supported in remote mode only. For procedural information in this guide, see [To List Log Levels](#).

## **list-loggers**

Lists all public loggers in your distribution of Eclipse GlassFish. Internal loggers are not listed. For procedural information in this guide, see [To List Loggers](#).

## **monitor**

Displays monitoring information for the common Eclipse GlassFish resources. Supported in remote mode only. For procedural information in this guide, see [To View Common Monitoring Data](#).

### **rotate-log**

Rotates the `server.log` file and stores the old data in a time-stamped file. Supported in remote mode only. For procedural information in this guide, see [To Rotate Log Files Manually](#).

### **set-log-attributes**

Sets log file attributes. Supported in remote mode only. For procedural information in this guide, see [Configuring the Logging Service](#).

### **set-log-file-format**

Sets the formatter used to format log records in log files. For procedural information in this guide, see [Setting the Log File Format](#).

### **set-log-levels**

Sets the log level for a module. Supported in remote mode only. For procedural information in this guide, see [Setting Log Levels](#).

## **ORB Subcommands**

### **create-iiop-listener**

Creates an IIOP listener. Supported in remote mode only. For procedural information in this guide, see [To Create an IIOP Listener](#).

### **delete-iiop-listener**

Deletes an IIOP listener. Supported in remote mode only. For procedural information in this guide, see [To Delete an IIOP Listener](#).

### **list-iiop-listeners**

Lists the existing IIOP listeners. Supported in remote mode only. For procedural information in this guide, see [To List IIOP Listeners](#).

## **Thread Pool Subcommands**

### **create-threadpool**

Creates a new thread pool. Supported in remote mode only. For procedural information in this guide, see [To Create a Thread Pool](#).

### **delete-threadpool**

Deletes the specified thread pool. Supported in remote mode only. For procedural information in this guide, see [To Delete a Thread Pool](#).

### **list-threadpools**

Lists the existing thread pools. Supported in remote mode only. For procedural information in this guide, see [To List Thread Pools](#).

# Transaction Service Subcommands

## **freeze-transaction-service**

Freezes the transaction subsystem during which time all the in-flight transactions are suspended. Supported in remote mode only. For procedural information, see [To Stop the Transaction Service](#).

## **recover-transactions**

Manually recovers pending transactions. Supported in remote mode only. For procedural information, see [To Manually Recover Transactions](#).

## **rollback-transaction**

Rolls back the named transaction. Supported in remote mode only. For procedural information, see [To Roll Back a Transaction](#).

## **unfreeze-transaction-service**

Resumes all the suspended in-flight transactions. Invoke this subcommand on an already frozen transaction. Supported in remote mode only. For procedural information, see [To Restart the Transaction Service](#).

# List of Examples

- [1-1 Determining if the DAS Requires Restart](#)
- [1-2 Determining if an Instance Requires Restart](#)
- [1-3 Listing Apache Felix Gogo Remote Shell Commands](#)
- [1-4 Running a Remote Shell Command](#)
- [1-5 Determining the Services That an OSGi Bundle Provides](#)
- [2-1 Running an asadmin Utility Subcommand in Single Mode](#)
- [2-2 Specifying an asadmin Utility Option With a Subcommand in Single Mode](#)
- [2-3 Specifying an asadmin Utility Option and a Subcommand Option in Single Mode](#)
- [2-4 Displaying Help Information for the asadmin Utility](#)
- [2-5 Displaying Help Information for an asadmin Utility Subcommand](#)
- [2-6 Starting a Multimode Session With asadmin Utility Options](#)
- [2-7 Starting a Multimode Session by Using the multimode Subcommand](#)
- [2-8 Running a Subcommand in a Multimode Session](#)
- [2-9 Running a Set of asadmin Subcommands From a File](#)
- [2-10 Using the --detach Option in Single Mode](#)
- [2-11 Using the --detach Option in Multimode](#)
- [2-12 Listing Jobs](#)
- [2-13 Attaching to a Subcommand and Checking Its Status](#)
- [2-14 Configuring Managed Jobs](#)
- [2-15 Creating a System Property](#)
- [2-16 Listing System Properties](#)
- [2-17 Deleting a System Property](#)
- [2-18 Adding Module Configuration to domain.xml](#)
- [2-19 Removing Module Configuration From domain.xml](#)
- [2-20 Displaying the Current Active Configuration of a Module](#)
- [2-21 Adding Resources](#)
- [2-22 Displaying Version Information](#)
- [2-23 Listing Applications](#)
- [2-24 Listing Containers](#)
- [2-25 Listing Modules](#)
- [2-26 Listing Subcommands](#)
- [2-27 Listing Timers](#)
- [2-28 Showing Status of a Component](#)

- 2-29 Determining the Methods and Method Parameters That an Object in the Tree Supports
- 2-30 Retrieving Data for an Object in the Tree
- 2-31 Adding an Object to the Tree
- 2-32 Updating an Object in the Tree
- 2-33 Deleting an Object From the Tree
- 3-1 Creating a Domain
- 3-2 Listing Domains
- 3-3 Logging In To a Domain on a Remote Machine
- 3-4 Logging In to a Domain on the Default Port of Localhost
- 3-5 Deleting a Domain
- 3-6 Starting a Domain
- 3-7 Stopping a Domain (or Server)
- 3-8 Restarting a Domain (or Server)
- 3-9 Restarting a Domain in a Browser
- 3-10 Creating a Service to Restart a DAS Automatically on Windows
- 3-11 Querying the Service to Restart a DAS Automatically on Windows
- 3-12 Creating a Service to Restart a DAS Automatically on Linux
- 3-13 Creating a Service to Restart a Domain Automatically on Oracle Solaris
- 3-14 Backing Up the Default Domain
- 3-15 Restoring the Default Domain
- 3-16 Listing Backups of the Default Domain
- 3-17 Displaying the DAS Uptime
- 3-18 Changing the Administration Port of a Domain
- 4-1 Creating JVM Options
- 4-2 Listing JVM Options
- 4-3 Deleting a JVM Option
- 4-4 Deleting Multiple JVM Options
- 4-5 Generating a JVM Report
- 4-6 Creating a Profiler
- 4-7 Deleting a Profiler
- 5-1 Creating a Thread Pool
- 5-2 Listing Thread Pools
- 5-3 Updating a Thread Pool
- 5-4 Deleting a Thread Pool
- 6-1 Invoking a Servlet With a URL

- 6-2 Invoking a Servlet From Within a JSP File
- 6-3 Redirecting a URL
- 6-4 httpd.conf File for mod\_jk
- 6-5 workers.properties File for mod\_jk
- 6-6 httpd.conf File for Load Balancing
- 6-7 workers.properties File for Load Balancing
- 6-8 http-ssl.conf File for mod\_jk Security
- 7-1 Changing the Name and Location of a Cluster's Log File
- 7-2 Listing Logger Levels for DAS
- 7-3 Listing Logger Levels for an Instance
- 7-4 Changing the Logger Log Level for a Cluster
- 7-5 Setting Log Levels for Multiple Loggers
- 7-6 Changing the Handler Log Level
- 7-7 Setting the Log File Format using set-log-file-format
- 7-8 Setting the Log File Format using set-log-attributes
- 7-9 Excluding Fields in the ODLLogFormatter
- 7-10 Excluding Fields in the GlassFishLogHandler
- 7-11 Disabling the Multiline Mode in the Log File
- 7-12 Changing the Rotation Size
- 7-13 Changing the Rotation Interval
- 7-14 Changing the Limit Number of Archived Log Files
- 7-15 Rotating Log Files Manually
- 7-16 Collecting and Downloading Log Files as a ZIP File
- 7-17 Listing Loggers
- 8-1 Enabling the Monitoring Service Dynamically
- 8-2 Enabling Monitoring for Modules Dynamically
- 8-3 Enabling Monitoring for Modules by Using the set Subcommand
- 8-4 Disabling the Monitoring Service Dynamically
- 8-5 Disabling Monitoring for Modules Dynamically
- 8-6 Disabling Monitoring by Using the set Subcommand
- 8-7 Viewing Common Monitoring Data
- 8-8 Viewing Attributes for a Specific Type
- 8-9 Viewing Monitorable Applications
- 8-10 Viewing Attributes for an Application
- 8-11 Viewing a Specific Attribute

- [9-1 Creating a Life Cycle Module](#)
- [9-2 Listing Life Cycle Modules](#)
- [9-3 Updating a Life Cycle Module](#)
- [9-4 Deleting a Life Cycle Module](#)
- [10-1 Listing Batch Jobs](#)
- [10-2 Listing Batch Job Executions](#)
- [10-3 Listing Batch Job Steps](#)
- [10-4 Listing the Batch Runtime Configuration](#)
- [10-5 Configuring the Batch Runtime](#)
- [11-1 Starting a Database](#)
- [11-2 Stopping a Database](#)
- [11-3 Creating a JDBC Connection Pool](#)
- [11-4 Listing JDBC Connection Pools](#)
- [11-5 Contacting a Connection Pool](#)
- [11-6 Resetting \(Flushing\) a Connection Pool](#)
- [11-7 Deleting a JDBC Connection Pool](#)
- [11-8 Creating a JDBC Resource](#)
- [11-9 Listing JDBC Resources](#)
- [11-10 Updating a JDBC Resource](#)
- [11-11 Deleting a JDBC Resource](#)
- [12-1 Creating a Connector Connection Pool](#)
- [12-2 Listing Connector Connection Pools](#)
- [12-3 Deleting a Connector Connection Pool](#)
- [12-4 Creating a Connector Resource](#)
- [12-5 Listing Connector Resources](#)
- [12-6 Deleting a Connector Resource](#)
- [12-7 Creating a Resource Adapter Configuration](#)
- [12-8 Listing Configurations for a Resource Adapter](#)
- [12-9 Deleting a Resource Adapter Configuration](#)
- [12-10 Creating a Connector Security Map](#)
- [12-11 Listing All Connector Security Maps for a Connector Connection Pool](#)
- [12-12 Listing Principals for a Specific Security Map for a Connector Connection Pool](#)
- [12-13 Listing Principals of All Connector Security Maps for a Connector Connection Pool](#)
- [12-14 Updating a Connector Security Map](#)
- [12-15 Deleting a Connector Security Map](#)

- 12-16 Creating Connector Work Security Maps
- 12-17 Listing the Connector Work Security Maps
- 12-18 Updating a Connector Work Security Map
- 12-19 Deleting a Connector Work Security Map
- 12-20 Creating an Administered Object
- 12-21 Listing Administered Objects
- 12-22 Deleting an Administered Object
- 13-1 Creating an HTTP Protocol
- 13-2 Listing the Protocols
- 13-3 Deleting a Protocol
- 13-4 Creating an HTTP Configuration
- 13-5 Deleting an HTTP Configuration
- 13-6 Creating a Transport
- 13-7 Listing HTTP Transports
- 13-8 Deleting a Transport
- 13-9 Creating an HTTP Listener
- 13-10 Creating a Network Listener
- 13-11 Listing HTTP Listeners
- 13-12 Updating an HTTP Network Listener
- 13-13 Deleting an HTTP Listener
- 13-14 Configuring an HTTP Listener for SSL
- 13-15 Deleting SSL From an HTTP Listener
- 13-16 Creating a Virtual Server
- 13-17 Listing Virtual Servers
- 13-18 Deleting a Virtual Server
- 14-1 Creating a Context Service
- 14-2 Listing Context Services
- 14-3 Deleting a Context Service
- 14-4 Creating a Managed Thread Factory
- 14-5 Listing Managed Thread Factories
- 14-6 Deleting a Managed Thread Factory
- 14-7 Creating a Managed Executor Service
- 14-8 Listing Managed Executor Services
- 14-9 Deleting a Managed Executor Service
- 14-10 Creating a Managed Scheduled Executor Service

- 14-11 Listing Managed Scheduled Executor Services
- 14-12 Deleting a Managed Scheduled Executor Service
- 15-1 Creating an IIOP Listener
- 15-2 Listing IIOP Listeners
- 15-3 Updating an IIOP Listener
- 15-4 Deleting an IIOP Listener
- 16-1 Creating a JavaMail Resource
- 16-2 Listing JavaMail Resources
- 16-3 Updating a JavaMail Resource
- 16-4 Deleting a JavaMail Resource
- 17-1 Creating a JMS Host
- 17-2 Listing JMS Hosts
- 17-3 Updating a JMS Host
- 17-4 Deleting a JMS Host
- 17-5 Creating a JMS Connection Factory
- 17-6 Creating a JMS Destination
- 17-7 Listing All JMS Resources
- 17-8 Listing a JMS Resources of a Specific Type
- 17-9 Deleting a JMS Resource
- 17-10 Creating a JMS Physical Destination
- 17-11 Listing JMS Physical Destinations
- 17-12 Flushing Messages From a JMS Physical Destination
- 17-13 Deleting a Physical Destination
- 18-1 Creating a Custom Resource
- 18-2 Listing Custom Resources
- 18-3 Updating a Custom JNDI Resource
- 18-4 Deleting a Custom Resource
- 18-5 Registering an External JNDI Resource
- 18-6 Listing JNDI Resources
- 18-7 Listing JNDI Entries
- 18-8 Updating an External JNDI Resource
- 18-9 Deleting an External JNDI Resource
- 19-1 Stopping the Transaction Service
- 19-2 Rolling Back a Transaction
- 19-3 Restarting the Transaction Service

- 19-4 Manually Recovering Transactions

# List of Figures

- [2-1 Web Page for the REST Resource for Managing a Domain](#)
- [2-2 Web Page for the REST Resource That Provides Class Loader Statistics](#)

# List of Tables

- [1-1 Default Administration Values](#)
- [1-2 Default Locations](#)
- [2-1 REST Resource Methods for Administering Monitoring and Configuration Data](#)
- [6-1 URL Fields for Servlets Within an Application](#)
- [8-1 HTTP Listener Common Monitoring Statistics](#)
- [8-2 JVM Common Monitoring Statistics](#)
- [8-3 Web Module Common Monitoring Statistics](#)
- [8-4 Example Resources Level Dotted Names](#)
- [8-5 EJB Cache Monitoring Statistics](#)
- [8-6 EJB Container Monitoring Statistics](#)
- [8-7 EJB Method Monitoring Statistics](#)
- [8-8 EJB Pool Monitoring Statistics](#)
- [8-9 Timer Monitoring Statistics](#)
- [8-10 HTTP Service Virtual Server Monitoring Statistics](#)
- [8-11 Jersey Statistics](#)
- [8-12 Connector Connection Pool Monitoring Statistics \(JMS\)](#)
- [8-13 Connector Work Management Monitoring Statistics \(JMS\)](#)
- [8-14 JVM Monitoring Statistics for Java SE Class Loading](#)
- [8-15 JVM Monitoring Statistics for Java SE - Threads](#)
- [8-16 JVM Monitoring Statistics for Java SE Compilation](#)
- [8-17 JVM Monitoring Statistics for Java SE Garbage Collectors](#)
- [8-18 JVM Monitoring Statistics for Java SE Memory](#)
- [8-19 JVM Statistics for the Java SE Operating System](#)
- [8-20 JVM Monitoring Statistics for Java SE Runtime](#)
- [8-21 Network Keep Alive Statistics](#)
- [8-22 Network Connection Queue Statistics](#)
- [8-23 Network File Cache Statistics](#)
- [8-24 Network Thread Pool Statistics](#)
- [8-25 ORB Monitoring Statistics \(Connection Manager\)](#)
- [8-26 General Resource Monitoring Statistics \(Connection Pool\)](#)
- [8-27 Application Specific Resource Monitoring Statistics \(Connection Pool\)](#)
- [8-28 EJB Security Monitoring Statistics](#)
- [8-29 Web Security Monitoring Statistics](#)

- [8-30 Realm Security Monitoring Statistics](#)
- [8-31 Thread Pool Monitoring Statistics](#)
- [8-32 JVM Monitoring Statistics for Java SE - Thread Info](#)
- [8-33 Transaction Service Monitoring Statistics](#)
- [8-34 Web Module Servlet Statistics](#)
- [8-35 Web JSP Monitoring Statistics](#)
- [8-36 Web Request Monitoring Statistics](#)
- [8-37 Web Servlet Monitoring Statistics](#)
- [8-38 Web Session Monitoring Statistics](#)
- [13-1 Default Ports for Listeners](#)
- [18-1 JNDI Lookup Names and Their Associated References](#)