Eclipse GlassFish Server Application Development Guide, Release 7

Eclipse GlassFish Server Application Development Guide, Release 7

Eclipse GlassFish Server

```
Application Development Guide

Release 7

Contributed 2018, 2019

This Application Development Guide describes how to create and run Java
Platform, Enterprise Edition (Java EE platform) applications that follow
the open Java standards model for Java EE components and APIs in the
Eclipse GlassFish Server environment. Topics include
developer tools, security, and debugging. This book is intended for use
by software developers who create, assemble, and deploy Java EE
applications using Oracle servers and software.

[[sthref1]]

'''''

Eclipse GlassFish Server Application Development 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.


[[GSDVG528]][[sthref2]]
```

```
[[preface]]
Preface
-------

[NOTE]
=======================================================================

This documentation is part of the Java Enterprise Edition contribution
to the Eclipse Foundation and is not intended for use in relation to
Java Enterprise Edition or Orace GlassFish. The documentation is in the
process of being revised to reflect the new Jakarta EE branding.
Additional changes will be made as requirements and procedures evolve
for Jakarta EE. Where applicable, references to Java EE or Java
Enterprise Edition should be considered references to Jakarta EE.

Please see the Title page for additional license information.
=======================================================================

This Application Development Guide describes how to create and run Java
Platform, Enterprise Edition (Java EE platform) applications that follow
the open Java standards model for Java EE components and APIs in the
Oracle GlassFish Server environment. Topics include developer tools,
security, and debugging. This book is intended for use by software
developers who create, assemble, and deploy Java EE applications using
Oracle servers and software.

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

GlassFish Server 5.0 is developed through the GlassFish project
open-source community at `https://javaee.github.io/glassfish/`. The
GlassFish project provides a structured process for developing the
GlassFish Server platform that makes the new features of the Java EE
platform available faster, while maintaining the most important feature
of Java EE: compatibility. It enables Java developers to access the
GlassFish Server source code and to contribute to the development of the
GlassFish Server. The GlassFish project is designed to encourage
communication between Oracle engineers and the community.

The following topics are addressed here:

* link:#ghpbz[GlassFish Server Documentation Set]
* link:#giprl[Related Documentation]
* link:#fwbkx[Typographic Conventions]
* link:#fquvc[Symbol Conventions]
* link:#ghpfg[Default Paths and File Names]
```

```
[[GSDVG00082]][[ghpbz]]


[[glassfish-server-documentation-set]]
GlassFish Server Documentation Set
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

[width="100%",cols="30%,70%",options="header",]
|=======================================================================
|Book Title |Description
|link:../release-notes/toc.html#GSRLN[Release Notes] |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.

|link:../quick-start-guide/toc.html#GSQSG[Quick Start Guide] |Explains how to get
started with the
GlassFish Server product.

|link:../installation-guide/toc.html#GSING[Installation Guide] |Explains how to
install the software
and its components.

|link:../administration-guide/toc.html#GSADG[Administration Guide] |Explains how to
configure, monitor,
and manage GlassFish Server subsystems and components from the command
line by using the link:../reference-manual/asadmin.html#GSRFM00263[`asadmin`]
utility. Instructions for
performing these tasks from the Administration Console are provided in
the Administration Console online help.

|link:../security-guide/toc.html#GSSCG[Security Guide] |Provides instructions for
configuring and
administering GlassFish Server security.

|link:../application-deployment-guide/toc.html#GSDPG[Application Deployment Guide]
|Explains how to assemble and
deploy applications to the GlassFish Server and provides information
about deployment descriptors.

|link:../application-development-guide/toc.html#GSDVG[Application Development Guide]
```

```
|Explains how to create and
implement Java Platform, Enterprise Edition (Java EE platform)
applications that are intended to run on the GlassFish Server. These
applications follow the open Java standards model for Java EE components
and application programmer interfaces (APIs). This guide provides
information about developer tools, security, and debugging.

| |

|link:../error-messages-reference/toc.html#GSEMR[Error Message Reference] |Describes
error messages that you
might encounter when using GlassFish Server.

|link:../reference-manual/toc.html#GSRFM[Reference Manual] |Provides reference
information in man
page format for GlassFish Server administration commands, utility
commands, and related concepts.

|link:../../openmq/mq-release-notes/toc.html#GMRLN[Message Queue Release Notes]
|Describes new features,
compatibility issues, and existing bugs for Open Message Queue.
|=======================================================================


[[GSDVG00083]][[giprl]]


[[related-documentation]]
Related Documentation
~~~~~~~~~~~~~~~~~~~~~

The following tutorials explain how to develop Java EE applications:

* https://javaee.github.io/firstcup/[Your First Cup: An Introduction to
the Java EE Platform] (`https://javaee.github.io/firstcup/`). For
beginning Java 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.
* https://javaee.github.io/tutorial/[The Java EE 8 Tutorial]
(`https://javaee.github.io/tutorial/`). This comprehensive tutorial
explains how to use Java EE 8 platform technologies and APIs to develop
Java EE applications.

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

```
* The Java EE specifications and API specification for version 8 is
located at `https://javaee.github.io/javaee-spec/`.
* The API specification for GlassFish Server 5.0, including Java EE 8
platform packages and nonplatform packages that are specific to the
GlassFish Server product, is located at
`https://javaee.github.io/glassfish/documentation`.

For information about creating enterprise applications in the NetBeans
Integrated Development Environment (IDE), see the
http://www.netbeans.org/kb/[NetBeans Documentation, Training & Support
page] (`http://www.netbeans.org/kb/`).

For information about the Apache Derby database for use with the
GlassFish Server, see:
(`http://www.oracle.com/technetwork/java/javadb/overview/index.html`).

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

[[GSDVG00084]][[fwbkx]]


[[typographic-conventions]]
Typographic Conventions
~~~~~~~~~~~~~~~~~~~~~~~

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

[width="100%",cols="14%,37%,49%",options="header",]
|=======================================================================
|Typeface |Meaning |Example
|`AaBbCc123` |The names of commands, files, and directories, and
onscreen computer output a|
Edit your `.login` file.

Use `ls` `a` to list all files.

`machine_name% you have mail.`

|`AaBbCc123` |What you type, contrasted with onscreen computer output a|
`machine_name%` `su`

`Password:`
```

```
|AaBbCc123 |A placeholder to be replaced with a real name or value |The
command to remove a file is `rm` filename.

|AaBbCc123 |Book titles, new terms, and terms to be emphasized (note
that some emphasized items appear bold online) a|
Read Chapter 6 in the User's Guide.

A cache is a copy that is stored locally.

Do not save the file.

|=========================================================================


[[GSDVG00085]][[fquvc]]


[[symbol-conventions]]
Symbol Conventions
~~~~~~~~~~~~~~~~~~

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

[width="100%",cols="10%,26%,28%,36%",options="header",]
|=========================================================================
|Symbol |Description |Example |Meaning
|`[ ]` |Contains optional arguments and command options. |`ls [-l]` |The
`-l` option is not required.

|`{ \| }` |Contains a set of choices for a required command option.
|`-d {y\|n}` |The `-d` option requires that you use either the `y`
argument or the `n` argument.

|`${ }` |Indicates a variable reference. |`${com.sun.javaRoot}`
|References the value of the `com.sun.javaRoot` 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.
|=========================================================================
```

```
[[GSDVG00086]][[ghpfg]]


[[default-paths-and-file-names]]
Default Paths and File Names
~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

[width="100%",cols="14%,34%,52%",options="header",]
|=======================================================================
|Placeholder |Description |Default Value
|as-install + a|
Represents the base installation directory for GlassFish Server.

In configuration files, as-install is represented as follows:

`${com.sun.aas.installRoot}`

 a|
Installations on the Oracle Solaris operating system, Linux operating
system, and Mac OS operating system:

user's-home-directory`/glassfish3/glassfish`

Installations on the Windows operating system:

SystemDrive`:\glassfish3\glassfish`

|as-install-parent + |Represents the parent of the base installation
directory for GlassFish Server. a|
Installations on the Oracle Solaris operating system, Linux operating
system, and Mac operating system:

user's-home-directory`/glassfish3`

Installations on the Windows operating system:

SystemDrive`:\glassfish3`

|domain-root-dir + |Represents the directory in which a domain is
created by default. |as-install`/domains/`

|domain-dir + a|
Represents the directory in which a domain's configuration is stored.

In configuration files, domain-dir is represented as follows:
```

```
`${com.sun.aas.instanceRoot}`

 |domain-root-dir`/`domain-name

|instance-dir + |Represents the directory for a server instance.
|domain-dir`/`instance-name
|=========================================================================




[[fvxzc]][[GSDVG00045]][[part-i]]

Part I +
--------

[[development-tasks-and-tools]]
Development Tasks and Tools
---------------------------




[[GSDVG00002]][[beaaq]]


[[setting-up-a-development-environment]]
1 Setting Up a Development Environment
--------------------------------------

This chapter gives guidelines for setting up an application development
environment in the Oracle GlassFish Server. Setting up an environment
for creating, assembling, deploying, and debugging your code involves
installing the mainstream version of the GlassFish Server and making use
of development tools. In addition, sample applications are available.

The following topics are addressed here:

* link:#beaar[Installing and Preparing the Server for Development]
* link:#beaas[High Availability Features]
* link:#beaat[Development Tools]
* link:#beabf[Sample Applications]

[[beaar]][[GSDVG00090]][[installing-and-preparing-the-server-for-development]]

Installing and Preparing the Server for Development
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

For more information about GlassFish Server installation, see the
link:../installation-guide/toc.html#GSING[GlassFish Server Open Source Edition
Installation Guide].

The following components are included in the full installation.

* JDK
* GlassFish Server core

** Java Platform, Standard Edition (Java SE) 8

** Java EE 8 compliant application server

** Administration Console

** `asadmin` utility

** Other development and deployment tools

** Open Message Queue software

** Apache http://db.apache.org/derby/manuals[Derby database]

** Load balancer plug-ins for web servers

The NetBeans Integrated Development Environment (IDE) bundles the
GlassFish edition of the GlassFish Server, so information about this IDE
is provided as well.

After you have installed GlassFish Server, you can further optimize the
server for development in these ways:

* Locate utility classes and libraries so they can be accessed by the
proper class loaders. For more information, see
link:class-loaders.html#beadj[Using the Common Class Loader].
* Set up debugging. For more information, see
link:debugging-apps.html#beafc[Debugging Applications].
* Configure the Virtual Machine for the Java platform (JVM software).
For more information, see "link:../administration-
guide/jvm.html#GSADG00007[Administering the Virtual
Machine for the Java Platform]" in GlassFish Server Open Source Edition
Administration Guide.

[[beaas]][[GSDVG00091]][[high-availability-features]]

High Availability Features
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~

High availability features such as load balancing and session failover
are discussed in detail in the link:../ha-administration-
guide/toc.html#GSHAG[GlassFish Server Open Source
Edition High Availability Administration Guide]. This book describes the
following features in the following sections:

* For information about HTTP session persistence, see
link:webapps.html#beahe[Distributed Sessions and Persistence].
* For information about checkpointing of the stateful session bean
state, see link:ejb.html#beaib[Stateful Session Bean Failover].
* For information about failover and load balancing for Java clients,
see link:java-clients.html#beakt[Developing Java Clients].
* For information about load balancing for message-driven beans, see
link:jms.html#beaop[Load-Balanced Message Inflow].

[[beaat]][[GSDVG00092]][[development-tools]]

Development Tools
~~~~~~~~~~~~~~~~~

The following general tools are provided with the GlassFish Server:

* link:#beaau[The `asadmin` Command]
* link:#beaav[The Administration Console]

The following development tools are provided with the GlassFish Server
or downloadable from Oracle:

* link:#beaba[The Migration Tool]
* link:#beaaw[The NetBeans IDE]

The following third-party tools might also be useful:

* link:#beabb[Debugging Tools]
* link:#beabc[Profiling Tools]

[[beaau]][[GSDVG00333]][[the-asadmin-command]]

The `asadmin` Command
^^^^^^^^^^^^^^^^^^^^^

The `asadmin` command allows you to configure a local or remote server
and perform both administrative and development tasks at the command
line. For general information about `asadmin`, see the
link:../reference-manual/toc.html#GSRFM[GlassFish Server Open Source Edition
Reference Manual].
```

```
The `asadmin` command is located in the as-install`/bin` directory. Type
`asadmin help` for a list of subcommands.

[[beaav]][[GSDVG00334]][[the-administration-console]]

The Administration Console
^^^^^^^^^^^^^^^^^^^^^^^^^^

The Administration Console lets you configure the server and perform
both administrative and development tasks using a web browser. For
general information about the Administration Console, click the Help
button in the Administration Console. This displays the GlassFish Server
online help.

To access the Administration Console, type `http://`host`:4848` in your
browser. The host is the name of the machine on which the GlassFish
Server is running. By default, the host is `localhost`. For example:

[source,oac_no_warn]
----
http://localhost:4848
----

[[beaba]][[GSDVG00337]][[the-migration-tool]]

The Migration Tool
^^^^^^^^^^^^^^^^^^

The Migration Tool converts and reassembles Java EE applications and
modules developed on other application servers. This tool also generates
a report listing how many files are successfully and unsuccessfully
migrated, with reasons for migration failure. For more information and
to download the Migration Tool, see
`http://java.sun.com/j2ee/tools/migration/index.html`.

[[beaaw]][[GSDVG00338]][[the-netbeans-ide]]

The NetBeans IDE
^^^^^^^^^^^^^^^^

The NetBeans IDE allows you to create, assemble, and debug code from a
single, easy-to-use interface. The GlassFish edition of the GlassFish
Server is bundled with the NetBeans 6.1 IDE. To download the NetBeans
IDE, see `http://www.netbeans.org`. This site also provides
documentation on how to use the NetBeans IDE with the bundled GlassFish
edition of the GlassFish Server.
```

```
You can also use the GlassFish Server with the Java Studio Enterprise
software, which is built on the NetBeans IDE. For more information, see
`http://developers.sun.com/jsenterprise/`.

[[beabb]][[GSDVG00340]][[debugging-tools]]

Debugging Tools
^^^^^^^^^^^^^^^

You can use several debugging tools with the GlassFish Server. For more
information, see link:debugging-apps.html#beafc[Debugging Applications].

[[beabc]][[GSDVG00341]][[profiling-tools]]

Profiling Tools
^^^^^^^^^^^^^^^

You can use several profilers with the GlassFish Server. For more
information, see link:debugging-apps.html#beafn[Profiling Tools].

[[beabf]][[GSDVG00093]][[sample-applications]]

Sample Applications
~~~~~~~~~~~~~~~~~~~

Sample applications that you can examine and deploy to the GlassFish
Server are included as part of the Java EE 8 SDK bundle. The samples are
also available from `https://github.com/javaee/glassfish-samples`.

Most GlassFish Server samples have the following directory structure:

* The `docs` directory contains instructions for how to use the sample.
* The `pom.xml` file defines Maven targets for the sample.
* The `src/` directory contains source code for the sample.




[[GSDVG00003]][[beade]]


[[class-loaders]]
2 Class Loaders
---------------

Understanding Oracle GlassFish Server class loaders can help you
determine where to place supporting JAR and resource files for your
modules and applications.
```

```
In a JVM implementation, the class loaders dynamically load a specific
Java class file needed for resolving a dependency. For example, when an
instance of `java.util.Enumeration` needs to be created, one of the
class loaders loads the relevant class into the environment.

The following topics are addressed here:

* link:#beadf[The Class Loader Hierarchy]
* link:#gfqpi[Delegation]
* link:#beadk[Using the Java Optional Package Mechanism]
* link:#beadg[Class Loader Universes]
* link:#gatej[Application-Specific Class Loading]
* link:#beadh[Circumventing Class Loader Isolation]


[NOTE]
=======================================================================

The Web Profile of the GlassFish Server supports the EJB 3.1 Lite
specification, which allows enterprise beans within web applications,
among other features. The full GlassFish Server supports the entire EJB
3.1 specification. For details, see
http://jcp.org/en/jsr/detail?id=318[JSR 318]
(`http://jcp.org/en/jsr/detail?id=318`).

=======================================================================


For information about class loader debugging, see
link:debugging-apps.html#gkpdk[Class Loader Debugging].

[[beadf]][[GSDVG00094]][[the-class-loader-hierarchy]]

The Class Loader Hierarchy
~~~~~~~~~~~~~~~~~~~~~~~~~~

Class loaders in the GlassFish Server runtime follow a delegation
hierarchy that is illustrated in the following figure and fully
described in link:#fvxzq[Table 2-1].

The following table describes the class loaders in the GlassFish Server.

[[GSDVG531]][[sthref4]][[fvxzq]]


Table 2-1 Oracle GlassFish Server Class Loaders
```

```
[width="100%",cols="20%,80%",options="header",]
|=======================================================================
|Class Loader |Description
|Bootstrap |The Bootstrap class loader loads the basic runtime classes
provided by the JVM software.

|Extension |The Extension class loader loads classes from JAR files
present in the system extensions directory, domain-dir`/lib/ext`. It is
parent to the Public API class loader. See link:#beadk[Using the Java
Optional Package Mechanism].

|Public API |The Public API class loader makes available all classes
specifically exported by the GlassFish Server runtime for use by
deployed applications. This includes, but is not limited to, Java EE
APIs and other Oracle APIs. It is parent to the Common class loader.

|Common |The Common class loader loads JAR files in the as-install`/lib`
directory, followed by JAR files in the domain-dir`/lib` directory.
Using domain-dir`/lib` is recommended whenever possible, and required
for custom login modules and realms. It is parent to the Connector class
loader. See link:#beadj[Using the Common Class Loader].

|Connector |The Connector class loader is a single class loader instance
that loads individually deployed connector modules, which are shared
across all applications. It is parent to the Applib class loader and the
LifeCycleModule class loader.

|LifeCycleModule |The LifeCycleModule class loader is created once per
lifecycle module. Each lifecycle module's classpath is used to construct
its own class loader. For more information on lifecycle modules, see
link:lifecycle-listeners.html#beamc[Developing Lifecycle Listeners].

|Applib a|
The Applib class loader loads the library classes, specified during
deployment, for a specific enabled module or Java EE application; see
link:#gatej[Application-Specific Class Loading]. One instance of this
class loader is present in each class loader universe; see
link:#beadg[Class Loader Universes]. It is parent to the Archive class
loader.

When multiple deployed applications use the same library, they share the
same instance of the library. One library cannot reference classes from
another library.

| |

|Archive |The Archive class loader loads classes from the WAR, EAR, and
JAR files or directories (for directory deployment) of applications or
```

```
modules deployed to the GlassFish Server. This class loader also loads
any application-specific classes generated by the GlassFish Server
runtime, such as stub classes or servlets generated by JSP pages.
|=========================================================================


In previous GlassFish Server versions, the JVM options provided
`classpath-prefix` and `classpath-suffix` attributes that made it
possible to add JAR files or directories either in front of, or after
the application server's system `classpath`. These options are not
present in GlassFish Server 6.0.

The `classpath-prefix` was typically used to substitute another package
for one of the GlassFish Server packages, for example if a newer one was
available. This same result can be achieved on a per-application basis with the
`--libraries` option for the `deploy` subcommand. For more information,
see the link:../reference-manual/deploy.html#GSRFM00114[`deploy`(1)] help page.
The Java Optional Package Mechanism does what `classpath-suffix` used to do. For more
information, see link:#beadk[Using the Java Optional Package Mechanism].

[[gfqpi]][[GSDVG00095]][[delegation]]

Delegation
~~~~~~~~~~

Note that the class loader hierarchy is not a Java inheritance
hierarchy, but a delegation hierarchy. In the delegation design, a class
loader delegates class loading to its parent before attempting to load a
class itself. If the parent class loader cannot load a class, the class
loader attempts to load the class itself. In effect, a class loader is
responsible for loading only the classes not available to the parent.
Classes loaded by a class loader higher in the hierarchy cannot refer to
classes available lower in the hierarchy.

The Java Servlet specification recommends that a web module's class
loader look in the local class loader before delegating to its parent.
You can make this class loader follow the delegation inversion model in
the Servlet specification by setting `delegate="false"` in the
`class-loader` element of the `glassfish-web.xml` file. It is safe to do
this only for a web module that does not interact with any other
modules. For details, see "link:../application-deployment-guide/dd-
elements.html#GSDPG00110[class-loader]" in GlassFish
Server Open Source Edition Application Deployment Guide.

The default value is `delegate="true"`, which causes a web module's
class loader to delegate in the same manner as the other class loaders.
You must use `delegate="true"` for a web application that accesses EJB
components or that acts as a web service client or endpoint. For details
```

```
about `glassfish-web.xml`, see the link:../application-deployment-
guide/toc.html#GSDPG[GlassFish Server Open
Source Edition Application Deployment Guide].

For a number of packages, including `java.*` and `javax.*`, symbol
resolution is always delegated to the parent class loader regardless of
the `delegate` setting. This prevents applications from overriding core
Java runtime classes or changing the API versions of specifications that
are part of the Java EE platform.

[[beadk]][[GSDVG00096]][[using-the-java-optional-package-mechanism]]

Using the Java Optional Package Mechanism
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Optional packages are packages of Java classes and associated native
code that application developers can use to extend the functionality of
the core platform.

To use the Java optional package mechanism, copy the JAR files into the
domain-dir`/lib/ext` directory, or use the `asadmin add-library` command
with the `--type ext` option, then restart the server. For more
information about the `asadmin add-library` command, see the GlassFish
Server Open Source Edition Reference Manual.

For more information, see
http://docs.oracle.com/javase/8/docs/technotes/guides/extensions/extensions.html[Opti
onal
Packages - An Overview]
(`http://docs.oracle.com/javase/8/docs/technotes/guides/extensions/extensions.html`)
and
http://download.oracle.com/javase/tutorial/ext/basics/load.html[Understanding
Extension Class Loading]
(`http://docs.oracle.com/javase/tutorial/ext/basics/load.html`).

[[gchif]][[GSDVG00097]][[using-the-endorsed-standards-override-mechanism]]

Using the Endorsed Standards Override Mechanism
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Endorsed standards handle changes to classes and APIs that are bundled
in the JDK but are subject to change by external bodies.

To use the endorsed standards override mechanism, copy the JAR files
into the domain-dir`/lib/endorsed` directory, then restart the server.

For more information and the list of packages that can be overridden,
see
```

```
http://docs.oracle.com/javase/8/docs/technotes/guides/standards/[Endorsed
Standards Override Mechanism]
(`http://docs.oracle.com/javase/8/docs/technotes/guides/standards/`).

[[beadg]][[GSDVG00098]][[class-loader-universes]]

Class Loader Universes
~~~~~~~~~~~~~~~~~~~~~~

Access to components within applications and modules installed on the
server occurs within the context of isolated class loader universes,
each of which has its own Applib and Archive class loaders.

* Application Universe - Each Java EE application has its own class
loader universe, which loads the classes in all the modules in the
application.
* Individually Deployed Module Universe - Each individually deployed EJB
JAR or web WAR has its own class loader universe, which loads the
classes in the module.

A resource such as a file that is accessed by a servlet, JSP, or EJB
component must be in one of the following locations:

* A directory pointed to by the Libraries field or `--libraries` option
used during deployment
* A directory pointed to by the `library-directory` element in the
`application.xml` deployment descriptor
* A directory pointed to by the application or module's classpath; for
example, a web module's classpath includes these directories: +
[source,oac_no_warn]
----
module-name/WEB-INF/classes
module-name/WEB-INF/lib
----

[[gatej]][[GSDVG00099]][[application-specific-class-loading]]

Application-Specific Class Loading
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

You can specify module- or application-specific library classes in one
of the following ways:

* Use the Administration Console. Open the Applications component, then
go to the page for the type of application or module. Select the Deploy
button. Type the comma-separated paths in the Libraries field. For
details, click the Help button in the Administration Console.
* Use the `asadmin deploy` command with the `--libraries` option and
```

```
specify comma-separated paths. For details, see the
link:../reference-manual/toc.html#GSRFM[GlassFish Server Open Source Edition
Reference Manual].
* Use the `asadmin add-library` command with the `--type app` option,
then restart the server. For details, see the link:../reference-
manual/toc.html#GSRFM[GlassFish
Server Open Source Edition Reference Manual].


[NOTE]
=======================================================================

None of these alternatives apply to application clients. For more
information, see link:java-clients.html#gjpjt[Using Libraries with
Application Clients].

=======================================================================


You can update a library JAR file using dynamic reloading or by
restarting (disabling and re-enabling) a module or application. To add
or remove library JAR files, you can redeploy the module or application.

Application libraries are included in the Applib class loader. Paths to
libraries can be relative or absolute. A relative path is relative to
domain-dir`/lib/applibs`. If the path is absolute, the path must be
accessible to the domain administration server (DAS). The GlassFish
Server automatically synchronizes these libraries to all remote cluster
instances when the cluster is restarted. However, libraries specified by
absolute paths are not guaranteed to be synchronized.


[TIP]
=======================================================================

You can use application-specific class loading to specify a different
XML parser than the default GlassFish Server XML parser.

You can also use application-specific class loading to access different
versions of a library from different applications.

=======================================================================


If multiple applications or modules refer to the same libraries, classes
in those libraries are automatically shared. This can reduce the memory
footprint and allow sharing of static information. However, applications
or modules using application-specific libraries are not portable. Other
```

```
ways to make libraries available are described in
link:#beadh[Circumventing Class Loader Isolation].

One library cannot reference classes from another library.

For general information about deployment, including dynamic reloading,
see the link:../application-deployment-guide/toc.html#GSDPG[GlassFish Server Open
Source Edition Application
Deployment Guide].


[NOTE]
=======================================================================

If you see an access control error message when you try to use a
library, you may need to grant permission to the library in the
`server.policy` file. For more information, see
link:securing-apps.html#beabz[Changing Permissions for an Application].

=======================================================================


[[beadh]][[GSDVG00100]][[circumventing-class-loader-isolation]]

Circumventing Class Loader Isolation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Since each application or individually deployed module class loader
universe is isolated, an application or module cannot load classes from
another application or module. This prevents two similarly named classes
in different applications or modules from interfering with each other.

To circumvent this limitation for libraries, utility classes, or
individually deployed modules accessed by more than one application, you
can include the relevant path to the required classes in one of these
ways:

* link:#beadj[Using the Common Class Loader]
* link:#gcrnt[Sharing Libraries Across a Cluster]
* link:#beadl[Packaging the Client JAR for One Application in Another
Application]

[[beadj]][[GSDVG00342]][[using-the-common-class-loader]]

Using the Common Class Loader
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

To use the Common class loader, copy the JAR files into the
```

```
domain-dir`/lib` or as-install`/lib` directory, or use the
`asadmin add-library` command with the `--type common` option, then
restart the server. For more information about the `asadmin add-library`
command, see the GlassFish Server Open Source Edition Reference Manual.

Using the Common class loader makes an application or module accessible
to all applications or modules deployed on servers that share the same
configuration. However, this accessibility does not extend to
application clients. For more information, see
link:java-clients.html#gjpjt[Using Libraries with Application Clients].

For example, using the Common class loader is the recommended way of
adding JDBC drivers to the GlassFish Server. For a list of the JDBC
drivers currently supported by the GlassFish Server, see the
link:../release-notes/toc.html#GSRLN[GlassFish Server Open Source Edition Release
Notes]. For
configurations of supported and other drivers, see
"link:../administration-guide/jdbc.html#GSADG00579[Configuration Specifics for JDBC
Drivers]" in
GlassFish Server Open Source Edition Administration Guide.

To activate custom login modules and realms, place the JAR files in the
domain-dir`/lib` directory, then restart the server.

[[gcrnt]][[GSDVG00343]][[sharing-libraries-across-a-cluster]]

Sharing Libraries Across a Cluster
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

To share libraries across a specific cluster, copy the JAR files to the
domain-dir`/config/`cluster-config-name`/lib` directory.

[[beadl]][[GSDVG00344]][[packaging-the-client-jar-for-one-application-in-another-
application]]

Packaging the Client JAR for One Application in Another Application
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

By packaging the client JAR for one application in a second application,
you allow an EJB or web component in the second application to call an
EJB component in the first (dependent) application, without making
either of them accessible to any other application or module.

As an alternative for a production environment, you can have the Common
class loader load the client JAR of the dependent application as
described in link:#beadj[Using the Common Class Loader]. Restart the
server to make the dependent application accessible to all applications
or modules deployed on servers that share the same configuration.
```

```
[[fvyab]][[GSDVG00048]][[to-package-the-client-jar-for-one-application-in-another-
application]]

To Package the Client JAR for One Application in Another Application
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

1.  Deploy the dependent application.
2.  Add the dependent application's client JAR file to the calling
application.
* For a calling EJB component, add the client JAR file at the same level
as the EJB component. Then add a `Class-Path` entry to the `MANIFEST.MF`
file of the calling EJB component. The `Class-Path` entry has this
syntax: +
[source,oac_no_warn]
----
Class-Path: filepath1.jar filepath2.jar ...
----
Each filepath is relative to the directory or JAR file containing the
`MANIFEST.MF` file. For details, see the Java EE specification.
* For a calling web component, add the client JAR file under the
`WEB-INF/lib` directory.
3.  If you need to package the client JAR with both the EJB and web
components, set `delegate="true"` in the `class-loader` element of the
`glassfish-web.xml` file. +
This changes the Web class loader so that it follows the standard class
loader delegation model and delegates to its parent before attempting to
load a class itself. +
For most applications, packaging the client JAR file with the calling
EJB component is sufficient. You do not need to package the client JAR
file with both the EJB and web components unless the web component is
directly calling the EJB component in the dependent application.
4.  Deploy the calling application. +
The calling EJB or web component must specify in its
`glassfish-ejb-jar.xml` or `glassfish-web.xml` file the JNDI name of the
EJB component in the dependent application. Using an `ejb-link` mapping
does not work when the EJB component being called resides in another
application. +
You do not need to restart the server.




[[GSDVG00004]][[beafc]]


[[debugging-applications]]
3 Debugging Applications
```

This chapter gives guidelines for debugging applications in the Oracle GlassFish Server.

The following topics are addressed here:

- Enabling Debugging
- JPDA Options
- Generating a Stack Trace for Debugging
- Application Client Debugging
- Open Message Queue Debugging
- Enabling Verbose Mode
- Class Loader Debugging
- GlassFish Server Logging
- Profiling Tools

Enabling Debugging

When you enable debugging, you enable both local and remote debugging. To start the server in debug mode, use the `--debug` option as follows:

```
asadmin start-domain --debug [domain-name]
```

You can then attach to the server from the Java Debugger (`jdb`) at its default Java Platform Debugger Architecture (JPDA) port, which is 9009. For example, for UNIX systems:

```
jdb -attach 9009
```

For Windows:

```
jdb -connect com.sun.jdi.SocketAttach:port=9009
```

For more information about the `jdb` debugger, see the following links:

- Java Platform Debugger Architecture - The Java Debugger: `http://java.sun.com/javase/technologies/core/toolsapis/jpda/`
- Java Platform Debugger Architecture - Connecting with JDB: `http://java.sun.com/javase/technologies/core/toolsapis/jpda/`

GlassFish Server debugging is based on the JPDA. For more information, see JPDA Options.

You can attach to the GlassFish Server using any JPDA compliant debugger, including that of NetBeans (`http://www.netbeans.org`), Java Studio Enterprise, JBuilder, Eclipse, and so on.

You can enable debugging even when the GlassFish Server is started without the `--debug` option. This is useful if you start the GlassFish Server from the Windows Start Menu, or if you want to make sure that debugging is always turned on.

To Set the Server to Automatically Start Up in Debug Mode ^^^^^^^^^^^^^^^^^^

1. Use the Administration Console. Select the JVM Settings component under the relevant configuration.
2. Check the Debug Enabled box.
3. To specify a different port (from 9009, the default) to use when attaching the JVM software to a debugger, specify `address=` port-number in the Debug Options field.
4. To add JPDA options, add any desired JPDA debugging options in Debug Options. See JPDA Options.

See Also

For details, click the Help button in the Administration Console from the JVM Settings page.

JPDA Options ~~~~

The default JPDA options in GlassFish Server are as follows:

```
-Xdebug -agentlib:transport=dt_socket,server=y,suspend=n,address=9009
```

For Windows, you can change `dt_socket` to `dt_shmem`.

If you substitute `suspend=y`, the JVM software starts in suspended mode and stays suspended until a debugger attaches to it. This is helpful if you want to start debugging as soon as the JVM software starts.

To specify a different port (from 9009, the default) to use when attaching the JVM software to a debugger, specify `address=`port-number.

You can include additional options. A list of JPDA debugging options is available at `http://java.sun.com/javase/technologies/core/toolsapis/jpda/`.

Generating a Stack Trace for Debugging ~~~~~~~~~~~~~~

To generate a Java stack trace for debugging, use the `asadmin generate-jvm-report --type=thread` command. The stack trace goes to the domain-dir`/logs/server.log` file and also appears on the command prompt screen. For more information about the `asadmin generate-jvm-report` command, see the GlassFish Server Open Source Edition Reference Manual.

Application Client Debugging

When the `appclient` script executes the `java` command to run the Application Client Container (ACC), which in turn runs the client, it includes on the command line the value of the `VMARGS` environment variable. You can set this variable to any suitable value. For example:

```
VMARGS=-agentlib:transport=dt_socket,server=y,suspend=y,address=8118
```

For debugging an application client, you should set suspend to `y` so you can connect the debugger to the client before any code has actually executed. Otherwise, the client may start running and execute past the point you want to examine.

You should use different ports for the server and client if you are debugging both concurrently. For details about setting the port, see JPDA Options.

You can also include JVM options in the `appclient` script directly. For information about the `appclient` script, see the GlassFish Server Open Source Edition Reference Manual.

The Application Client Container is supported only in the full GlassFish Server, not in the Web Profile. See Developing Java Clients.

Open Message Queue Debugging

Open Message Queue has a broker logger, which can be useful for debugging Java Message Service (JMS) applications, including message-driven bean applications. You can adjust the logger's verbosity, and you can send the logger output to the broker's console using the broker's `-tty` option. For more information, see the Open Message Queue Administration Guide.

JMS resources are supported only in the full GlassFish Server, not in the Web Profile. See Using the Java Message Service.

Enabling Verbose Mode

To have the server logs and messages printed to `System.out` on your command prompt screen, you can start the server in verbose mode. This makes it easy to do simple debugging using print

statements, without having to view the `server.log` file every time.

To start the server in verbose mode, use the `--verbose` option as follows:

```
asadmin start-domain --verbose [domain-name]
```

When the server is in verbose mode, messages are logged to the console or terminal window in addition to the log file. In addition, pressing Ctrl-C stops the server and pressing Ctrl-\ (on UNIX platforms) or Ctrl-Break (on Windows platforms) prints a thread dump. On UNIX platforms, you can also print a thread dump using the `jstack` command (see `http://docs.oracle.com/javase/8/docs/technotes/tools/share/jstack.html`) or the command `kill -QUIT` process_id.

Class Loader Debugging

To generate class loading messages, use the following `asadmin create-jvm-options` command:

```
asadmin create-jvm-options -verbose\:class
```

To send the JVM messages to a special JVM log file instead of `stdout`, use the following `asadmin create-jvm-options` commands:

```
asadmin create-jvm-options -XX\:+LogVMOutput
asadmin create-jvm-options -XX\:LogFile=${com.sun.aas.instanceRoot}/logs/jvm.log
```

These `-XX` options are specific to the OpenJDK (or Hotspot) JVM and do not work with the JRockit JVM.

To send the GlassFish Server messages to the Administration Console instead of `stderr`, start the domain in verbose mode as described in Enabling Verbose Mode.

GlassFish Server Logging

You can use the GlassFish Server's log files to help debug your applications. Use the Administration Console. Select the Stand-Alone Instances component, select the instance from the table, then click the View Log Files button in the General Information page. Or select the Cluster component, select the cluster from the table, select the Instances tab, select the instance from the table, then click the View Log Files button in the General Information page.

To change logging settings, select Logger Settings under the relevant configuration.

For details about logging, click the Help button in the Administration Console.

Profiling Tools

You can use a profiler to perform remote profiling on the GlassFish Server to discover bottlenecks in server-side performance. This section describes how to configure profilers for use with GlassFish Server.

The following topics are addressed here:

- The NetBeans Profiler
- The HPROF Profiler
- The JProbe Profiler

Information about comprehensive monitoring and management support in the Java 2 Platform, Standard Edition (J2SE platform) is available at http://docs.oracle.com/javase/8/docs/technotes/guides/management/index.html.

The NetBeans Profiler

For information on how to use the NetBeans profiler, see http://profiler.netbeans.org/index.html.

The HPROF Profiler

The Heap and CPU Profiling Agent (HPROF) is a simple profiler agent shipped with the Java 2 SDK. It is a dynamically linked library that interacts with the Java Virtual Machine Profiler Interface (JVMPI) and writes out profiling information either to a file or to a socket in ASCII or binary format.

HPROF can monitor CPU usage, heap allocation statistics, and contention profiles. In addition, it can also report complete heap dumps and states of all the monitors and threads in the Java virtual machine. For more details on the HPROF profiler, see the technical article at http://java.sun.com/developer/technicalArticles/Programming/HPROF.html.

After HPROF is enabled using the following instructions, its libraries are loaded into the server process.

To Use HPROF Profiling on UNIX

Deployment directories may change between GlassFish Server releases.

An alternative way to add permissions to a specific application or module is to edit the `granted.policy` file for that application or module. The `granted.policy` file is located in the domain-dir`/generated/policy/`app-or-module-name directory. In this case, you add permissions to the default grant block. Do not delete permissions from this file.

When the GlassFish Server policy subsystem determines that a permission should not be granted, it logs a `server.policy` message specifying the permission that was not granted and the protection domains, with indicated code source and principals that failed the protection check. For example, here is the first part of a typical message:

```
[#|2005-12-17T16:16:32.671-0200|INFO|sun-appserver-pe9.1|
javax.enterprise.system.core.security|_ThreadID=14;_ThreadName=Thread-31;|
JACC Policy Provider: PolicyWrapper.implies, context(null)-
permission((java.util.PropertyPermission java.security.manager write))
domain that failed(ProtectionDomain
(file:/E:/glassfish/domains/domain1/applications/cejug-clfds/ ... )
...
```

Granting the following permission eliminates the message:

```
grant codeBase "file:${com.sun.aas.instanceRoot}/applications/cejug-clfds/-" {
    permission java.util.PropertyPermission "java.security.manager", "write";
}
```

Do not add `java.security.AllPermission` to the `server.policy` file for application code. Doing so completely defeats the purpose of the security manager, yet you still get the performance overhead associated with it.

As noted in the Java EE specification, an application should provide documentation of the additional permissions it needs. If an application requires extra permissions but does not document the set it needs, contact the application author for details.

As a last resort, you can iteratively determine the permission set an application needs by observing `AccessControlException` occurrences in the server log.

If this is not sufficient, you can add the `-Djava.security.debug=failure` JVM option to the domain. Use the following `asadmin create-jvm-options` command, then restart the server:

```
asadmin create-jvm-options -Djava.security.debug=failure
```

For more information about the `asadmin create-jvm-options` command, see the GlassFish Server Open Source Edition Reference Manual.

You can use the J2SE standard `policytool` or any text editor to edit the `server.policy` file. For more information, see `http://docs.oracle.com/javase/tutorial/security/tour2/index.html`.

For detailed information about policy file syntax, see `http://docs.oracle.com/javase/8/docs/technotes/guides/security/PolicyFiles.html`.

For information about using system properties in the `server.policy` file, see `http://docs.oracle.com/javase/8/docs/technotes/guides/security/PolicyFiles.html`.

For detailed information about the permissions you can set in the `server.policy` file, see `http://docs.oracle.com/javase/8/docs/technotes/guides/security/permissions.html`.

The Javadoc for the `Permission` class is at `http://docs.oracle.com/javase/8/docs/api/java/security/Permission.html`.

Enabling and Disabling the Security Manager ^^^^^^^^^^^^^^^

The security manager is disabled by default.

In a production environment, you may be able to safely disable the security manager if all of the following are true:

- Performance is critical
- Deployment to the production server is carefully controlled
- Only trusted applications are deployed
- Applications don't need policy enforcement

Disabling the security manager may improve performance significantly for some types of applications.

To enable the security manager, do one of the following:

- To use the Administration Console, open the Security component under the relevant configuration, and check the Security Manager Enabled box. Then restart the server. For details, click the Help button in the Administration Console.
- Use the following `asadmin create-jvm-options` command, then restart the server:

```
asadmin create-jvm-options -Djava.security.manager
```

To disable the security manager, uncheck the Security Manager Enabled box or use the

corresponding `asadmin delete-jvm-options` command. For more information about `create-jvm-options` and `delete-jvm-options`, see the GlassFish Server Open Source Edition Reference Manual.

If the security manager is enabled and you are using the Java Persistence API by calling `Persistence.createEMF()`, the EclipseLink persistence provider requires that you set the `eclipselink.security.usedoprivileged` JVM option to `true` as follows:

```
asadmin create-jvm-options -Declipselink.security.usedoprivileged=true
```

If the security manager is enabled and you are using the Java Persistence API by injecting or looking up an entity manager or entity manager factory, the EJB container sets this JVM option for you.

You must grant additional permissions to CDI-enabled Java EE applications that are deployed in a GlassFish Server 5.0 domain or cluster for which security manager is enabled. These additional permissions are not required when security manager is disabled.

To deploy CDI-enabled Java EE applications in a GlassFish Server 5.0 domain or cluster for which security manager is enabled, add the following permissions to the applications:

```
grant codeBase "file:${com.sun.aas.instanceRoot}/applications/[ApplicationName]" {
 permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};
```

For example, for a CDI application named `foo.war`, add the following permissions to the `server.policy` file, restart the domain or cluster, and then deploy and use the application.

```
grant codeBase "file:${com.sun.aas.instanceRoot}/applications/foo" {
 permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};
```

For more information about modifying application permissions, see Changing Permissions for an Application.

Configuring Message Security for Web Services

In message security, security information is applied at the message layer and travels along with the web services message. Web Services Security (WSS) is the use of XML Encryption and XML Digital Signatures to secure messages. WSS profiles the use of various security tokens including X.509 certificates, Security Assertion Markup Language (SAML) assertions, and username/password tokens to achieve this.

Message layer security differs from transport layer security in that it can be used to decouple message protection from message transport so that messages remain protected after transmission, regardless of how many hops they travel.

Message security (JSR 196) is supported only in the full GlassFish Server, not in the Web Profile.

In this release of the GlassFish Server, message layer annotations are not supported.

For more information about web services, see Developing Web Services.

For more information about message security, see the following:

- "https://javaee.github.io/tutorial/security-intro.html[Introduction to Security in the Java EE Platform]" in The Java EE 8 Tutorial
- GlassFish Server Open Source Edition Security Guide
- JSR 196 (`http://www.jcp.org/en/jsr/detail?id=196`), Java Authentication Service Provider Interface for Containers
- The Liberty Alliance Project specifications at `http://www.projectliberty.org/resources/specifications.php/?f=resources/specifications.php`
- The Oasis Web Services Security (WSS) specification at `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`
- The Web Services Interoperability Organization (WS-I) Basic Security Profile (BSP) specification at `http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html`
- The XML and Web Services Security page at `http://xwss.java.net/`
- The WSIT page at `http://wsit.java.net/`

The following topics are addressed here:

- Message Security Providers
- Message Security Responsibilities
- Application-Specific Message Protection
- Understanding and Running the Sample Application

Message Security Providers ^^^^^^^^^^

When you first install the GlassFish Server, the providers `XWS_ClientProvider` and `XWS_ServerProvider` are configured but disabled. You can enable them in one of the following ways:

- To enable the message security providers using the Administration Console, open the Security

component under the relevant configuration, select the Message Security component, and select SOAP. Then select `XWS_ServerProvider` from the Default Provider list and `XWS_ClientProvider` from the Default Client Provider list. For details, click the Help button in the Administration Console.

- You can enable the message security providers using the following commands.

```
asadmin set
server-config.security-service.message-security-
config.SOAP.default_provider=XWS_ServerProvider
asadmin set
server-config.security-service.message-security-
config.SOAP.default_client_provider=XWS_ClientProvider
```

For more information about the `asadmin set` command, see the GlassFish Server Open Source Edition Reference Manual.

The example described in Understanding and Running the Sample Application uses the `ClientProvider` and `ServerProvider` providers, which are enabled when the Ant targets are run. You don't need to enable these on the GlassFish Server prior to running the example.

If you install the OpenSSO, you have these additional provider choices:

- `AMClientProvider` and `AMServerProvider` - These providers secure web services and Simple Object Access Protocol (SOAP) messages using either WS-I BSP or Liberty ID-WSF tokens. These providers are used automatically if they are configured as the default providers. If you wish to override any provider settings, you can configure these providers in `message-security-binding` elements in the `glassfish-web.xml`, `glassfish-ejb-jar.xml`, and `glassfish-application-client.xml` deployment descriptor files.
- `AMHttpProvider` - This provider handles the initial end user authentication for securing web services using Liberty ID-WSF tokens and redirects requests to the OpenSSO for single sign-on. To use this provider, specify it in the `httpservlet-security-provider` attribute of the `glassfish-web-app` element in the `glassfish-web.xml` file.

Liberty specifications can be viewed at `http://www.projectliberty.org/resources/specifications.php/?f=resources/specifications.php`. The WS-I BSP specification can be viewed at `http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html`.

For more information about the GlassFish Server deployment descriptor files, see the GlassFish Server Open Source Edition Application Deployment Guide.

For information about configuring these providers in the GlassFish Server, see the GlassFish Server Open Source Edition Security Guide. For additional information about overriding provider settings, see Application-Specific Message Protection.

You can create new message security providers in one of the following ways:

- To create a message security provider using the Administration Console, open the Security component under the relevant configuration, and select the Message Security component. For details, click the Help button in the Administration Console.
- You can use the `asadmin create-message-security-provider` command to create a message security provider. For details, see the GlassFish Server Open Source Edition Reference Manual.

In addition, you can set a few optional provider properties using the `asadmin set` command. For example:

```
asadmin set server-config.security-service.message-security-config.provider-
config.property.debug=true
```

The following table describes these message security provider properties.

Table 4-2 Message Security Provider Properties

Property	Default	Description
`security.config`	domain-dir`/config/wss-server-`config-1.0.xml`	Specifies the location of the message security configuration file. To point to a configuration file in the domain-dir`/config` directory, use the system property `${com.sun.aas.instanceRoot}/``config/`, for example: `${com.sun.aas.instanceRoot}/config/``wss-server-config-1.0.xml` See System Properties.
`debug`	`false`	If `true`, enables dumping of server provider debug messages to the server log.
`dynamic.username.password`	`false`	If `true`, signals the provider runtime to collect the user name and password from the `CallbackHandler` for each request. If `false`, the user name and password for `wsse:UsernameToken(s)` is collected once, during module initialization. This property is only applicable for a `ClientAuthModule`.
`encryption.key.alias`	`s1as`	Specifies the encryption key used by the provider. The key is identified by its `keystore` alias.

Property	Default	Description
`signature.key.alias`	`s1as`	Specifies the signature key used by the provider. The key is identified by its `keystore` alias.

Message Security Responsibilities ^^^^^^^^^^

In the GlassFish Server, the system administrator and application deployer roles are expected to take primary responsibility for configuring message security. In some situations, the application developer may also contribute, although in the typical case either of the other roles may secure an existing application without changing its implementation and without involving the developer.

The following topics are addressed here:

- Application Developer Responsibilities
- Application Deployer Responsibilities
- System Administrator Responsibilities

Application Developer Responsibilities