Eclipse GlassFish Application Deployment Guide, Release 7

Eclipse GlassFish

Application Deployment Guide

Release 7

Contributed 2018 - 2024

This Application Deployment Guide describes deployment of applications and application components to Eclipse GlassFish, and includes information about deployment descriptors.

Preface

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 Jakarta EE or Java Enterprise Edition should be considered references to Jakarta EE.

Please see the Title page for additional license information.

This Application Deployment Guide describes deployment of applications and application components to Eclipse GlassFish, and includes information about deployment descriptors.

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
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.
Quick Start Guide	Explains how to get started with the Eclipse GlassFish product.

Book Title	Description
Installation Guide	Explains how to install the software and its components.
Upgrade Guide	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.
Deployment Planning Guide	Explains how to build a production deployment of Eclipse GlassFish that meets the requirements of your system and enterprise.
Administration Guide	Explains how to configure, monitor, and manage Eclipse GlassFish subsystems and components from the command line by using the `asadmin`(1M) utility. Instructions for performing these tasks from the Administration Console are provided in the Administration Console online help.
Security Guide	Provides instructions for configuring and administering Eclipse GlassFish security.
Application Deployment Guide	Explains how to assemble and deploy applications to the Eclipse GlassFish and provides information about deployment descriptors.
Application Development Guide	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.
Add-On Component Development Guide	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.
Embedded Server Guide	Explains how to run applications in embedded Eclipse GlassFish and to develop applications in which Eclipse GlassFish is embedded.
High Availability Administration Guide	Explains how to configure Eclipse GlassFish to provide higher availability and scalability through failover and load balancing.
Performance Tuning Guide	Explains how to optimize the performance of Eclipse GlassFish.
Troubleshooting Guide	Describes common problems that you might encounter when using Eclipse GlassFish and explains how to solve them.
Error Message Reference	Describes error messages that you might encounter when using Eclipse GlassFish.
Reference Manual	Provides reference information in man page format for Eclipse GlassFish administration commands, utility commands, and related concepts.
Message Queue Release Notes	Describes new features, compatibility issues, and existing bugs for Open Message Queue.

Book Title	Description
Message Queue Technical Overview	Provides an introduction to the technology, concepts, architecture, capabilities, and features of the Message Queue messaging service.
Message Queue Administration Guide	Explains how to set up and manage a Message Queue messaging system.
Message Queue Developer's Guide for JMX Clients	Describes the application programming interface in Message Queue for programmatically configuring and monitoring Message Queue resources in conformance with the Java Management Extensions (JMX).
Message Queue Developer's Guide for Java Clients	Provides information about concepts and procedures for developing Java messaging applications (Java clients) that work with Eclipse GlassFish.
Message Queue Developer's Guide for C Clients	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 `.login` file. Use `ls a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`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)	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.	`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.

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: `${com.sun.aas.installRoot}`	• Installations on the Oracle Solaris operating system, Linux operating system, and Mac OS operating system: user's-home-directory`/glassfish7/glassfish` • Installations on the Windows operating system: SystemDrive`:\glassfish7\glassfish`
as-install-parent	Represents the parent of the base installation directory for Eclipse GlassFish.	• Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: user's-home-directory`/glassfish7` • Installations on the Windows operating system: SystemDrive`:\glassfish7`
domain-root-dir	Represents the directory in which a domain is created by default.	as-install`/domains/`
domain-dir	Represents the directory in which a domain's configuration is stored. In configuration files, domain-dir is represented as follows: `${com.sun.aas.instanceRoot}`	domain-root-dir/domain-name
instance-dir	Represents the directory for a server instance.	domain-dir/instance-name

1 Overview of Eclipse GlassFish 7 Application Deployment

Eclipse GlassFish 7 provides an environment for developing and deploying Java applications and web services. Eclipse GlassFish applications include Java Platform, Enterprise Edition (Jakarta EE platform) standard features as well as features specific to Eclipse GlassFish. This guide explains the tools and processes used for deploying applications and modules in the Eclipse GlassFish environment. Only Eclipse GlassFish features are described in detail in this document.

The following topics are addressed here:

- About Application Deployment
- About Assembly and Deployment Events
- About Deployment Tools
- Additional Information on Application Deployment

Information and instructions on deploying from the command line are provided in this document. Information and instructions for accomplishing the deployment tasks by using the Administration Console are contained in the Administration Console online help.

About Application Deployment

Assembly, also known as packaging, is the process of combining discrete components of an application or module into a single unit that can be installed on an application server. The Eclipse GlassFish assembly process conforms to the customary Jakarta EE specifications. The only difference is that when you assemble applications or modules in Eclipse GlassFish, you can include optional Eclipse GlassFish deployment descriptors that enhance functionality.

Deployment is the process of installing an application or module on Eclipse GlassFish, optionally specifying location-specific information, such as a list of local users that can access the application, or the name of the local database. Eclipse GlassFish deployment tools expand the archive file into an open directory structure that is ready for users. Eclipse GlassFish deployment tools are described in About Deployment Tools.

The following topics are addressed here:

- General Deployment Functionality
- Deployment Descriptors and Annotations
- Modules and Applications
- Access to Shared Framework Classes
- Naming Standards
- Module and Application Versions

General Deployment Functionality

Various Jakarta EE module types, such as connector module, web module, EJB module, application client module, can be deployed in the following ways:

- Archive Deployment. Deploys the application as an archive file. For instructions, see To Deploy an Application or Module.
- Dynamic Reloading. Redeploys the application by creating or modifying a special `.reload` file in the applications repository. For instructions, see To Reload Changes to Applications or Modules Dynamically.
- Automatic Deployment. Deploys the application archive that is placed in the autodeployment directory. For instructions, see To Deploy an Application or Module Automatically.
- Directory Deployment. Deploys the application in a directory format. For instructions, see To Deploy an Application or Module in a Directory Format.
- JSR 88 Deployment. A deployment mechanism implemented based on the JSR 88 standard from `jcp.org`. It delivers vendor neutral deployment options. See JSR 88 Client.

A deployment plan, which deploys a portable archive along with a deployment plan containing Eclipse GlassFish deployment descriptors, can apply to any of these deployment techniques. For instructions, see To Deploy an Application or Module by Using a Deployment Plan.

There are two work situations that require different safeguards and processes:

- A development environment provides a loose set of tools and work spaces for a relatively small number of developers who are creating and testing applications and modules.
- A production environment provides a stable, protected environment where applications are tuned to maximum efficiency for business use rather than for development.

Some deployment methods that are used effectively in a development environment should not be used in production. In addition, whenever a reload is done, the sessions that are in transit become invalid, which might not be a concern for development, but can be a serious matter in production. The client must restart the session, another negative in a production environment.

For production environments, any upgrade should be performed as a rolling upgrade, which upgrades applications and modules without interruption in service. For more information, see Upgrading Applications Without Loss of Availability in Eclipse GlassFish High Availability Administration Guide.

Deployment Descriptors and Annotations

A deployment descriptor is an XML file that describes how a Jakarta EE application or module should be deployed. Each deployment descriptor XML file has a corresponding Document Type Definition (DTD) file or schema (XSD) file, which defines the elements, data, and attributes that the deployment descriptor file can contain. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options, and also describes specific configuration requirements that you must resolve.

Because the information in a deployment descriptor is declarative, it can be changed without

requiring modifications to source code. During deployment, Eclipse GlassFish reads the information in the deployment descriptor and deploys the application or module as directed.

The following types of deployment descriptors are associated with Eclipse GlassFish:

- Jakarta EE Standard Descriptors. Jakarta EE standard deployment descriptors are described in the Jakarta EE specification. You can find the specification at `https://jakarta.ee/specifications/`. Information about the XML schemas that define Jakarta EE standard deployment descriptors is available at `https://jakarta.ee/xml/ns/jakartaee/`.

  The Jakarta EE specification permits the use of alternate top-level standard deployment descriptors that reside outside of the application archive using the `alt-dd` mechanism (alternate module-level deployment descriptors were permitted prior to Jakarta EE 7). Alternate deployment descriptors are described in the Jakarta EE specification. You can find the specification at `https://jakarta.ee/specifications/platform/`. Alternate deployment descriptors override the top-level deployment descriptors packaged in an application archive. For example, for EAR files, an alternate deployment descriptor overrides `application.xml`. For standalone modules, an alternate deployment descriptor overrides the top-level module descriptor, such as `web.xml`.

- Eclipse GlassFish Descriptors. Eclipse GlassFish provides optional deployment descriptors for configuring features that are specific to Eclipse GlassFish. For example, when you assemble an EJB module, you annotate or create two Eclipse GlassFish deployment descriptor files with these names: `ejb-jar.xml` and `glassfish-ejb-jar.xml`. If the EJB component is an entity bean with container-managed persistence (CMP), you can also create a `.dbschema` file and a `sun-cmp-mapping.xml` file. For complete descriptions of these files and their elements, see Eclipse GlassFish Deployment Descriptor Files and Elements of the Eclipse GlassFish Deployment Descriptors.

  Eclipse GlassFish also permits the use of alternate top-level Eclipse GlassFish runtime deployment descriptors that reside outside of an application archive. Alternate Eclipse GlassFish deployment descriptors override the top-level deployment descriptors packaged in the archive. For example, for EAR files, an alternate Eclipse GlassFish deployment descriptor overrides `glassfish-application.xml`. For standalone modules, an alternate Eclipse GlassFish deployment descriptor overrides the top-level module descriptor, such as `glassfish-web.xml`. The name of the Eclipse GlassFish alternate deployment descriptor file must begin with `glassfish-`. Alternate deployment descriptors do not apply to `sun-*.xml` deployment descriptors.

  Unless otherwise stated, settings in the Eclipse GlassFish deployment descriptors override corresponding settings in the Jakarta EE standard descriptors and in the Eclipse GlassFish configuration.

An annotation, also called metadata, enables a declarative style of programming. You can specify information within a class file by using annotations. When the application or module is deployed, the information can either be used or overridden by the deployment descriptor. Eclipse GlassFish supports annotation according to the following specifications:

- JSR 250 Common Annotation Specification
- JSR 181 Annotation for Web Services Specification

- EJB 3.1 Specification

The following annotation and deployment descriptor combinations are supported:

- Jakarta EE applications or modules can be packaged with full Jakarta EE compliant standard and runtime deployment descriptors. If the standard deployment descriptors have specified the `metadata-complete` attribute, annotations in the application or module are ignored.
- Jakarta EE applications or modules can be fully annotated with metadata defined by the listed specifications. Annotation eliminates the need for Jakarta EE standard deployment descriptors. In most cases, the Eclipse GlassFish deployment descriptors are also not needed.
- Jakarta EE applications or modules can be partially annotated with some deployment information in standard deployment descriptors. In case of conflicts, deployment descriptor values supersede the annotated metadata, and a warning message is logged.

Modules and Applications

An application is a logical collection of one or more modules joined by application annotations or deployment descriptors. You assemble components into JAR, WAR, or RAR files, then combine these files and, optionally, deployment descriptors into an Enterprise archive (EAR) file which is deployed.

A module is a collection of one or more Jakarta EE components that run in the same container type, such as a web container or EJB container. The module uses annotations or deployment descriptors of that container type. You can deploy a module alone or as part of an application.

The following topics are addressed here:

- Types of Modules
- Module-Based Deployment
- Application-Based Deployment

Types of Modules

Eclipse GlassFish supports the following types of modules:

- Web Module. A web module, also known as a web application, is a collection of servlets, EJBs, HTML pages, classes, and other resources that you can bundle and deploy to several Jakarta EE application servers. A web application archive (WAR) file is the standard format for assembling web applications. A WAR file can consist of the following items: servlets, JavaServer Pages (JSP) files, JSP tag libraries, utility classes, static pages, client-side applets, beans, bean classes, enterprise bean classes, plus annotations or web deployment descriptors (`web.xml` and `glassfish-web.xml`).
- EJB Module. An EJB module is a deployable software unit that consists of one or more enterprise beans, plus an EJB deployment descriptor. A Java archive (JAR) file is the standard format for assembling enterprise beans. An EJB JAR file contains the bean classes (home, remote, local, and implementation), all of the utility classes, and annotations or deployment descriptors (`ejb-jar.xml` and `glassfish-ejb-jar.xml`). If the EJB component is a version 2.1 or earlier entity bean with container managed persistence (CMP), you can also include a `.dbschema` file and a CMP

mapping descriptor (`sun-cmp-mapping.xml`).

- Connector Module. A connector module, also known as a resource adapter module, is a deployable software unit that provides a portable way for EJB components to access foreign enterprise information system (EIS) data. A connector module consists of all Java interfaces, classes, and native libraries for implementing a resource module, plus a resource deployment descriptor. A resource adapter archive (RAR) is the standard format for assembling connector modules. Each Eclipse GlassFish connector has annotations or a deployment descriptor file (`ra.xml`).

  After deploying a J2EE connector module, you must configure it as described in Developing Connectors in Eclipse GlassFish Application Development Guide.

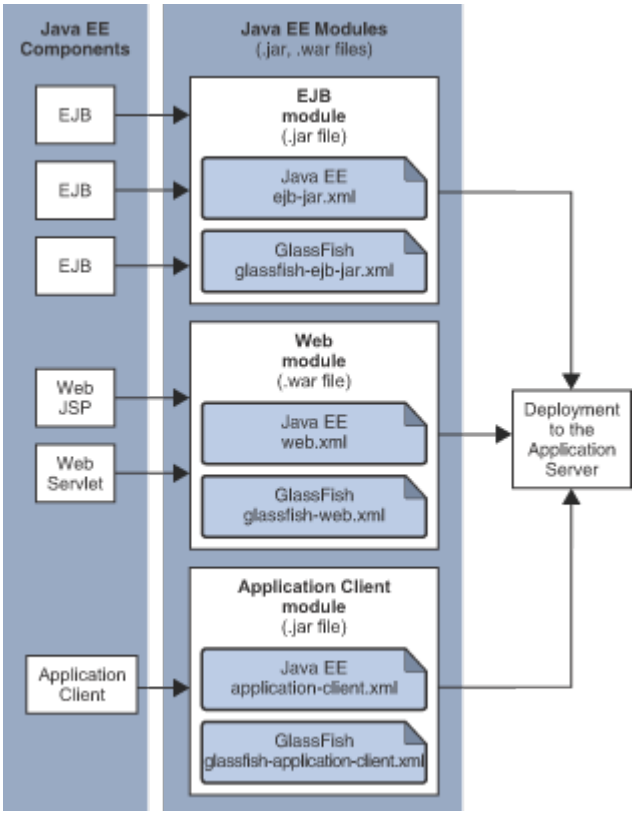
- Application Client Module. An application client module is a deployable software unit that consists of one or more classes, and application client deployment descriptors (`application-client.xml` and `glassfish-application-client.xml`). An application client JAR file applies to a Eclipse GlassFish type of Jakarta EE client. An application client supports the standard Jakarta EE Application Client specifications.
- Lifecycle Module. A lifecycle module provides a means of running short-duration or long-duration Java-based tasks within the Eclipse GlassFish environment. Lifecycle modules are not Jakarta EE standard modules. See Developing Lifecycle Listeners in Eclipse GlassFish Application Development Guide for more information.

Module-Based Deployment

You can deploy web, EJB, and application client modules separately, outside of any application. Module-based deployment is appropriate when components need to be accessed by other modules, applications, or application clients. Module-based deployment allows shared access to a bean from a web, EJB, or application client component.

The following figure shows separately-deployed EJB, web, and application client modules.

Figure 1-1 Module-Based Assembly and Deployment

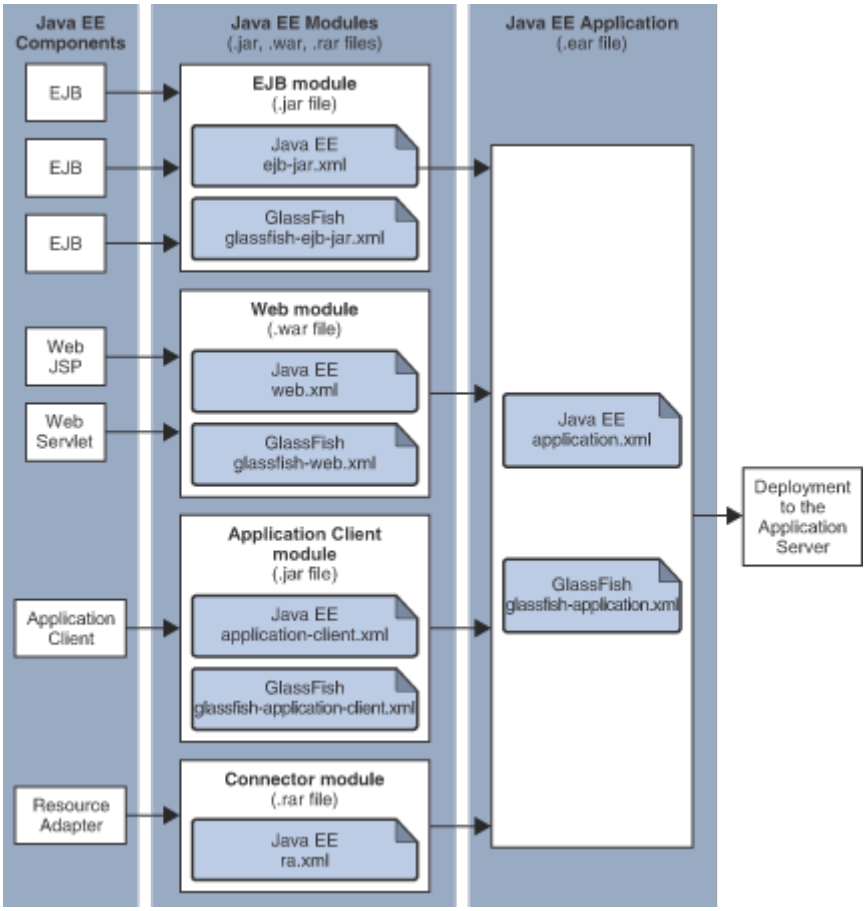


Application-Based Deployment

Application-based deployment is appropriate when components need to work together as one unit.

The following figure shows EJB, web, application client, and connector modules assembled into a Jakarta EE application.

Figure 1-2 Application-Based Assembly and Deployment



Access to Shared Framework Classes

If you assemble a large, shared library into every module that uses it, the result is a huge file that takes too long to register with the server. In addition, several versions of the same class could exist in different class loaders, which is a waste of resources. When Jakarta EE applications and modules use shared framework classes (such as utility classes and libraries), the classes can be put in the path for the common class loader or an application-specific class loader rather than in an application or module.

To specify an application-specific library file during deployment, use the `--libraries` option of the `deploy` or `redeploy` subcommand of the `asadmin` command. To add a library JAR file to the Common class loader directory, the Java optional package directory, or the application-specific class loader directory, use the `add-library` subcommand. You can then list the libraries with `list-libraries` and remove the libraries with `remove-library`. For more information about all these commands, see the Eclipse GlassFish Reference Manual.

For more information about class loaders, see Class Loaders in Eclipse GlassFish Application Development Guide.

According to the Jakarta EE specification, section 8.1.1.2, "Dependencies," you cannot package utility classes within an individually-deployed EJB module. Instead, you must package the EJB module and utility JAR within an application using the JAR Extension Mechanism Architecture.

Naming Standards

Names of applications and individually-deployed modules must be unique within a Eclipse GlassFish domain. Modules within an application must have unique names. In addition, for enterprise beans that use container-managed persistence (CMP), the `.dbschema` file names must be unique within an application.

You should use a hierarchical naming scheme for module file names, EAR file names, module names as found in the `module-name` portion of the `ejb-jar.xml` files, and EJB names as found in the `ejb-name` portion of the `ejb-jar.xml` files. This hierarchical naming scheme ensures that name collisions do not occur. The benefits of this naming practice apply not only to Eclipse GlassFish, but to other Jakarta EE application servers as well.

The following topics are addressed here:

- Portable Naming
- JNDI Naming
- Directory Structure

Portable Naming

Starting in Jakarta EE 6, the Jakarta EE specification defines the portable `application-name`, which allows you to specify an application name in the `application.xml` file. For example:

```
<application-name>xyz</application-name>
```

The Jakarta EE specification also defines the portable `module-name` element in the module standard deployment descriptors.

Eclipse GlassFish determines the application registration name according to the following order of precedence:

1. The name specified at deployment time in the Administration Console or in the `--name` option of the `asadmin deploy` command is used.
2. If no name is specified at deployment time, the portable `application-name` or `module-name` in the Jakarta EE deployment descriptor is used.
3. If no name is specified at deployment time or in the deployment descriptors, the archive name, minus the file type suffix, is used.

JNDI Naming

Java Naming and Directory Interface (JNDI) lookup names for EJB components must also be unique. Establishing a consistent naming convention can help. For example, appending the application name and the module name to the EJB name is a way to guarantee unique names, such as, `jms/qConnPool`.

Directory Structure

Application and module directory structures must follow the structure outlined in the Jakarta EE specification. During deployment, the application or module is expanded from the archive file to an open directory structure. The directories that hold the individual modules are named with `_jar`, `_rar`, and `_war` suffixes.

If you deploy a directory instead of an EAR file, your directory structure must follow this same convention. For instructions on performing directory deployment, see To Deploy an Application or Module in a Directory Format.

Module and Application Versions

Application and module versioning allows multiple versions of the same application to exist in a Eclipse GlassFish domain, which simplifies upgrade and rollback tasks. At most one version of an application or module can be enabled on a server any given time. Versioning provides extensions to tools for deploying, viewing, and managing multiple versions of modules and applications, including the Administration Console and deployment-related `asadmin` subcommands. Different versions of the same module or application can have the same context root or JNDI name. Use of versioning is optional.

The following topics are addressed here:

- Version Identifiers and Expressions
- Choosing the Enabled Version
- Versioning Restrictions and Limitations

Version Identifiers and Expressions

The version identifier is a suffix to the module or application name. It is separated from the name by a colon (`:`). It must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (`-`), and period (`.`) characters. The following examples show valid version identifiers for the `foo` application:

```
foo:1
foo:BETA-2e
foo:3.8
foo:patch39875
```

A module or application without a version identifier is called the untagged version. This version can coexist with other versions of the same module or application that have version identifiers.

In some deployment-related `asadmin` commands, you can use an asterisk (`*`) as a wildcard character to specify a version expression, which selects multiple version identifiers. Using the asterisk by itself after the colon selects all versions of a module or application, including the untagged version. The following table shows example version expressions and the versions they select.

Version Expression	Selected Versions
`foo:*`	All versions of `foo`, including the untagged version
`foo:BETA*`	All `BETA` versions of `foo`
`foo:3.*`	All `3.`x versions of `foo`
`foo:patch*`	All `patch` versions of `foo`

The following table summarizes which `asadmin` subcommands are identifier-aware or expression-aware. All expression-aware subcommands are also identifier-aware.

Identifier-Aware Subcommands	Expression-Aware Subcommands
`deploy`, `deploydir`, `redeploy`	`undeploy`
`enable`	`disable`
`list-sub-components`	`show-component-status`
`get-client-stubs`	`create-application-ref`, `delete-application-ref`

The `create-application-ref` subcommand is expression-aware only if the `--enabled` option is set to `false`. Because the `--enabled` option is set to `true` by default, the `create-application-ref` subcommand is identifier-aware by default.

The `list-applications` and `list-application-refs` subcommands display information about all deployed versions of a module or application. To find out which version is enabled, use the `--long` option.

Choosing the Enabled Version

At most one version of a module or application can be enabled on a server instance. All other versions are disabled. Enabling one version automatically disables all others. You can disable all versions of a module or application, leaving none enabled.

The `--enabled` option of the `deploy` and `redeploy` subcommands is set to `true` by default. Therefore, simply deploying or redeploying a module or application with a new version identifier enables the new version and disables all others. To deploy a new version in a disabled state, set the `--enabled` option to `false`.

To enable a version that has been deployed previously, use the `enable` subcommand.

Versioning Restrictions and Limitations

Module and application versioning in Eclipse GlassFish is subject to the following restrictions and limitations:

- Use of the `--name` option is mandatory for modules and applications that use versioning. There is no automatic version identifier generation.
- Eclipse GlassFish does not recognize any relationship between versions such as previous or later versions. All version relationships must be tracked manually.
- There is no limit to the number of versions you can deploy except what is imposed by disk space

limits.

- A module or application in a directory should not be deployed twice with a different version identifier. To redeploy a module or application from a directory with a new version, you must use the `--force` option of the `deploy` subcommand.
- Database tables created or deleted as part of deployment and undeployment are global resources and cannot be qualified by an application version. Be very careful when using global resources among versions of the same application.
- Web sessions are preserved during redeployment of a new version. However, preserving sessions among different versions of the same module or application is complex, because the key used for session variables is the same for the old and new versions.
- Resources are created with reference to a resource-adapter's module or application name. This means that an older version's resources do not automatically refer to a newer version of the module or application. Therefore, you must explicitly create resources for a newer version of a module or application. Eclipse GlassFish ignores duplicate exported global resources and lets deployment succeed.
- OSGi already has its own versioning system. Therefore, when you deploy an OSGi bundle, Eclipse GlassFish ignores any version information provided with the name but permits the deployment to succeed with warnings.

About Assembly and Deployment Events

The deployment tools that are provided by Eclipse GlassFish can be used by any user authorized as an administrator to deploy applications and modules into any Eclipse GlassFish environment. However, effective application deployment requires planning and care. Only the developer knows exactly what is required by an application, so the developer is responsible for initial assembly and deployment.

1. Deployment Descriptor or Annotation Creation. The developer creates the deployment descriptors or equivalent annotations using Java standards and tools.

   Details of the Eclipse GlassFish deployment descriptors are contained in Eclipse GlassFish Deployment Descriptor Files and Elements of the Eclipse GlassFish Deployment Descriptors. The Eclipse GlassFish sample applications contain deployment descriptors that can be used as templates for developing deployment descriptors.

2. Assembly. The developer assembles the archive file(s) using Java standards and tools, such as the `jar` command. The application or module is packaged into a JAR, WAR, RAR, or EAR file. For guidelines on naming, see Naming Standards.

   There are no Eclipse GlassFish issues to consider.

3. Test Deployment. The developer performs a test deployment of the archive. For instructions, see To Deploy an Application or Module.
4. Archive Submission. The developer submits the verified archive to the administrator for deployment into a production environment. The developer includes instructions for any additional deployment tasks that the administrator must perform. For an example of such

additional instructions, see Access to Shared Framework Classes.

5. Configuration. The administrator applies additional deployment specifics. Sometimes the developer has indicated additional deployment needs, such as specifying the production database. In this case, the administrator edits and reassembles the archive.
6. Production Deployment. The administrator deploys the archive to production. See To Deploy an Application or Module.
7. Troubleshooting. If deployment fails, the administrator returns the archive to the developer. The developer fixes the problem and resubmits the archive to the administrator. Sometimes the administrator resolves the problem, depending on what the problem is.

About Deployment Tools

Eclipse GlassFish provides tools for assembling and deploying a module or application.

The following topics are addressed here:

- Administration Console
- The `asadmin` Utility
- NetBeans IDE
- Eclipse IDE
- JSR 88 Client

Administration Console

The Eclipse GlassFish Administration Console is a browser-based utility that features a graphical interface that includes extensive online help for the administrative tasks. The format for starting the Administration Console in a web browser is `http://`hostname`:`port. For example:

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

Step-by-step instructions for using the Administration Console for deployment are provided in the Administration Console online help. You can display the help material for a page by clicking the Help button. The initial help page describes the functions and fields of the page itself. To find instructions for performing associated tasks, click a link in the See Also list.

The `asadmin` Utility

The Eclipse GlassFish `asadmin` utility is a command-line tool that invokes subcommands for identifying the operation or task that you want to perform. You can run `asadmin` commands either from a command prompt or from a script. The format for starting the `asadmin` utility on the command line is as-install`/bin/asadmin` subcommand --option. For example:

```
asadmin list-applications --type web
```

Application deployment commands are listed in The `asadmin` Deployment Subcommands. All Eclipse GlassFish `asadmin` subcommands are documented in the Eclipse GlassFish Reference Manual.

For the most part, you can perform the same administrative tasks by using either the graphical Administration Console or the `asadmin` command-line utility, however, there are exceptions. Procedures for using the command-line utilities are provided in this guide and in the command-line help pages, which are similar to man pages. You can display the help material for a command by typing help followed by the subcommand. For example:

```
asadmin help list-applications
```

For additional information on the `asadmin` utility, see "Using the `asadmin` Utility" in Eclipse GlassFish Administration Guide and the `asadmin`(1M) help page.

NetBeans IDE

You can use the NetBeans Integrated Development Environment (IDE), or another IDE, to assemble Jakarta EE applications and modules. For additional information, see `https://netbeans.apache.org/`.

Eclipse IDE

In addition to the NetBeans IDE, a plug-in for the Eclipse IDE extends GlassFish to the Eclipse community.

JSR 88 Client

The syntax of the URI entry for the `getDeploymentManager` method is as follows:

```
deployer:Sun:AppServer::admin-host:admin-port[:https]
```

For example:

```
deployer:Sun:AppServer::localhost:4848:https
```

Additional Information on Application Deployment

As specified from Jakarta EE specifications, the relevant specifications are the following:

- Jakarta EE Platform, Enterprise Edition 10 Specification
  `https://jakarta.ee/specifications/platform/`
- Jakarta EE Application Deployment JSR 88 Specification
  `http://jcp.org/en/jsr/detail?id=88`
- Common Annotations for the Java Platform 1.6 Specification
  `http://jcp.org/en/jsr/detail?id=250`

- Java Servlet 3.0 Specification
  http://jcp.org/en/jsr/detail?id=315
- Enterprise JavaBeans 3.1 Specification
  http://jcp.org/en/jsr/detail?id=318
- Jakarta EE Connector Architecture 1.6 Specification
  http://jcp.org/en/jsr/detail?id=322

The following product documentation might be relevant to some aspects of application deployment:

- Eclipse GlassFish Application Development Guide
- Eclipse GlassFish Administration Guide
- Eclipse GlassFish Add-On Component Development Guide
- Eclipse GlassFish Reference Manual
- Eclipse GlassFish Administration Console online help

2 Deploying Applications

This chapter provides procedures and guidelines for deploying applications and modules in the Eclipse GlassFishenvironment by using the `asadmin` command-line utility.

The following topics are addressed here:

- Deploying Applications and Modules
- Modifying the Configuration of a Web Application or Module
- Web Module Deployment Guidelines
- EJB Module Deployment Guidelines
- Deploying a Connector Module
- Assembling and Deploying an Application Client Module
- Lifecycle Module Deployment Guidelines
- Web Service Deployment Guidelines
- OSGi Bundle Deployment Guidelines
- Transparent JDBC Connection Pool Reconfiguration
- Application-Scoped Resources

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

Deploying Applications and Modules

Application deployment is a dynamic process, which means that deployed applications and modules become available without requiring you to restart the server instance. Dynamic deployment can be useful in production environments to bring new applications and modules online easily. If you do restart the server, all deployed components are still deployed and available.

The following topics are addressed here:

- To Deploy an Application or Module
- To Change Targets for a Deployed Application or Module
- To List Deployed Applications or Modules
- To Redeploy an Application or Module
- To Disable an Application or Module
- To Enable an Application or Module
- To Undeploy an Application or Module
- To Reload Changes to Applications or Modules Dynamically
- To Deploy an Application or Module Automatically
- To Deploy an Application or Module by Using a Deployment Plan

- To Deploy an Application or Module in a Directory Format

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

To Deploy an Application or Module

Use the `deploy` subcommand in remote mode to deploy an assembled application or module to Eclipse GlassFish. If an error occurs during deployment, the application or module is not deployed. If a module within an application contains an error, the entire application is not deployed. These failures prevent a partial deployment that could leave the server in an inconsistent state.

By default, the deployment target is the default server instance, `server`. To deploy only to the default server instance, specify no target. If you deploy the application or module only to the domain target, it exists in the domain central repository, but no server instances or clusters can reference the component unless you add references.

You can also deploy a component to a specific stand-alone server instance or cluster. When you deploy to server instances or clusters, the application or module exists in the domain's central repository and is referenced by any clusters or server instances that you deployed to. For a cluster, the preselected deployment target is `server`.

If the component is already deployed or already exists, you can forcefully redeploy if you set the `--force` option of the `deploy` subcommand to true. The `redeploy` subcommand also accomplishes this. See Example 2-10. You can see the enabled or disabled status of an application or module by using the `show-component-status` subcommand.

For information about how the application or module name is derived, see Naming Standards.

Use the `--altdd` or `--runtimealtdd` options of the `deploy` (and `redeploy`) subcommand to deploy an application or module using a top-level alternate deployment descriptor. The `--altdd` option specifies a top-level alternate Jakarta EE standard deployment descriptor. The `--runtimealtdd` option specifies a top-level alternate Eclipse GlassFish runtime deployment descriptor. See Example 2-3. For more information about deployment descriptors associated with Eclipse GlassFish, see Deployment Descriptors and Annotations.

You can also specify the deployment order of an application by using the `--deploymentorder` option of the `deploy` (and `redeploy`) subcommand. This is useful for applications that must be loaded in a certain order at server startup. Applications with lower deployment order numbers are loaded first. See Example 2-4. If a deployment order is not specified at the time an application is deployed, the default deployment order of 100 is assigned. If two applications have the same deployment order, the application that was deployed first is loaded first at server startup.

1. Ensure that the server is running.

   Remote commands require a running server.

2. List deployed applications by using the `list-applications` subcommand.
3. Deploy the application or module by using the `deploy` subcommand.

Information about the options and properties of the subcommand is included in this help page.

4. If needed, fix issues and rerun the `deploy` subcommand.

Example 2-1 Deploying an Enterprise Application

This example deploys `newApp.ear` to the default server, `server`.

```
asadmin> deploy Cart.ear
Application deployed successfully with name Cart.
Command deploy executed successfully
```

Example 2-2 Deploying a Connector Module

This example deploys a connector module that is packaged in an RAR file.

```
asadmin> deploy jdbcra.rar
Application deployed successfully with name jdbcra.
Command deploy executed successfully
```

Example 2-3 Using an Alternate Jakarta EE Standard Deployment Descriptor

This example deploys an application using an alternate Jakarta EE standard deployment descriptor file that resides outside of an application archive. Specify an absolute path or a relative path to the alternate deployment descriptor file.

```
asadmin> deploy --altdd path_to_alternate_descriptor cart.ear
Application deployed successfully with name cart.
Command deploy executed successfully
```

Example 2-4 Specifying the Deployment Order of an Application

This example specifies the deployment order of two applications. The `cart` application is loaded before the `horse` application at server startup.

Some lines of output are omitted from this example for readability.

```
asadmin> deploy --deploymentorder 102 --name cart cart.war
...
asadmin> deploy --deploymentorder 110 --name horse horse.war
...
```

See Also

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

To Change Targets for a Deployed Application or Module

After deployment, the deployed application or module exists in the central repository and can be referenced by the server instances or clusters that you deployed to as targets. The `asadmin create-application-ref` and `asadmin delete-application-ref` subcommands enable you to add or delete targets for a deployed component. Because the application or module itself is stored in the central repository, adding or deleting targets adds or deletes the same version of the component on different targets.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Add and remove targets by using the `create-application-ref` and `delete-application-ref` subcommands.

See Also

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

To List Deployed Applications or Modules

There are a number of commands that can be used to list deployed applications or modules and their subcomponents. Use the commands in this section in remote mode.

1. Ensure that the server is running.

   Remote commands require a running server.

2. List the desired applications by using the `list-applications` subcommand or the `list-sub-components` subcommand.

   Information about these commands is included in these help pages.

3. Show the status of a deployed component by using the `show-component-status` subcommand.

Example 2-5 Listing Applications

The `list-applications` subcommand lists all deployed Jakarta EE applications or modules. If the `--type` option is not specified, all components are listed. This example lists deployed applications.

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

Example 2-6 Listing Subcomponents

The `list-sub-components` subcommand lists EJBs or servlets in a deployed module or in a module of the deployed application. If a module is not identified, all modules are listed. The `--appname` option

functions only when the given module is standalone. To display a specific module in an application, you must specify the module name and the `--appname` option. This example gets the subcomponents of module `mejb.jar` within application `MEjbApp`.

```
asadmin> list-sub-components --appname MEjbApp mejb.jar
MEJBBean <StatelessSessionBean>
Command list-sub-components executed successfully
```

Example 2-7 Showing Status of a Deployed Component

The `show-component-status` subcommand gets the status (enabled or disabled) of the deployed component. This example gets the status of the `MEjbApp` component.

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

To Redeploy an Application or Module

Use the `redeploy` subcommand in remote mode to overwrite a previously-deployed application or module. You can also accomplish this task by using the `--force` option of the `deploy` subcommand. Whenever a redeployment is done, the HTTP and SFSB sessions in transit at that time, and the EJB timers, become invalid unless you use the `--keepstate=true` option of the `redeploy` subcommand.

Before You Begin

You must remove a preconfigured resource before it can be updated.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Redeploy an application or module by using the `redeploy` subcommand or the `deploy` subcommand with the `--force` option.

   Information about the options and properties of these commands is included in these help pages.

Example 2-8 Retaining HTTP Session State During Redeployment

This example redeploys the `hello` web application. In a production environment, you usually want to retain sessions. If you use the `--keepstate` option, active sessions of the application are retained and restored when redeployment is complete.

```
asadmin> redeploy --name hello --keepstate=true hello.war
Application deployed successfully with name hello.
Command redeploy executed successfully.
```

Keep State is a checkbox option when you redeploy using the Administration Console. For instructions, see the Administration Console online help.

Example 2-9 Redeploying a Web Application That Was Deployed From a Directory

This example redeploys the `hello` web application, which was originally deployed from the `hellodir` directory.

```
asadmin>redeploy --name hellodir
Application deployed successfully with name hellodir.
Command redeploy executed successfully.
```

Example 2-10 Redeploying an Application by Using `asadmin deploy --force`

The `--force` option is set to `false` by default. This example redeploys `newApp.ear` even if has been deployed or already exists.

```
asadmin> deploy --force=true newApp.ear
Application deployed successfully with name newApp.
Command deploy executed successfully.
```

See Also

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

To Disable an Application or Module

Use the `disable` subcommand in remote mode to immediately deactivate a deployed application or module without removing it from the server. Disabling a component makes the component inaccessible to clients. However, the component is not overwritten or uninstalled, and can be enabled by using the `asadmin enable` subcommand.

An application or module is enabled by default.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Obtain the exact name of the application or module that you are disabling.

   To list deployed applications or modules, use the `list-applications` subcommand. If you do not specify a type, all deployed applications and modules are listed. For example, valid types can be `web`, `ejb`, `connector`, `application`, and `webservice`.

   To see the status of deployed components, use the `show-component-status` subcommand.

3. Deactivate the application or module by using the `disable` subcommand.

Information about the options and properties of the subcommand is included in this help page.

Example 2-11 Listing Deployed Web Applications

This example lists all deployed web applications.

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

Example 2-12 Disabling a Web Application

This example disables the `hellojsp` application.

```
asadmin> disable hellojsp
Command disable executed successfully.
```

See Also

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

To Enable an Application or Module

An enabled application or module is runnable and can be accessed by clients if it has been deployed to an accessible server instance or cluster. An application or module is enabled by default. Use the `enable` subcommand in remote mode to enable an application or module that has been disabled.

An application or module that is deployed to more than one target can be enabled on one target and disabled on another. If a component is referenced by a target, it is not available to users unless it is enabled on that target.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Enable the application or module by using the `enable` subcommand.

   If the component has not been deployed, an error message is displayed. If the component is already enabled, it is re-enabled. To see the status of deployed components, use the `show-component-status` subcommand.

   Information about the options and properties of the subcommand is included in this help page.

Example 2-13 Enabling an Application

This example enables the `sampleApp` application.

```
asadmin> enable sampleApp
Command enable executed successfully.
```

See Also

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

To Undeploy an Application or Module

Use the `undeploy` subcommand in remote mode to uninstall a deployed application or module and remove it from the repository. To reinstate the component, you must deploy the component again using the `deploy` subcommand.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Obtain the exact name of the application or module you are undeploying.

   To list deployed applications or modules, use the `list-applications` subcommand. If you do not specify a type, all deployed applications and modules are listed. For example, valid types can be `web`, `ejb`, `connector`, `application`, and `webservice`.

   To see the status of deployed components, use the `show-component-status` subcommand.

3. Undeploy the application or module by using the `undeploy` subcommand.

   Information about the options and properties of the subcommand is included in this help page.

Example 2-14 Listing Deployed Applications or Modules

This example lists all applications of type `web`.

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

Example 2-15 Undeploying an Application

This example uninstalls the `hellojsp` application.

```
asadmin> undeploy hellojsp
hellojsp <web>
Command undeploy executed successfully.
```

See Also

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

To Reload Changes to Applications or Modules Dynamically

Dynamic reloading enables you to change the code or deployment descriptors of an application or module without needing to perform an explicit redeployment. Instead, you can copy the changed class files or descriptors into the deployment directory for the application or module. The server checks for changes periodically and automatically redeploys the changes if the timestamp of the `.reload` file in the root directory for the application or module has changed.

Dynamic reloading is enabled by default, and is available only on the default server instance.

1. Go to the root directory of the deployed application or module.

   For an application:

   ```
   domain-dir/applications/app-name
   ```

   For an individually deployed module:

   ```
   domain-dir/applications/module-name
   ```

   Deployment directories might change between Eclipse GlassFish releases.

2. Create or update the timestamp of the `.reload` file to load the changes.
   - For UNIX: `touch .reload`
   - For Windows: `echo> .reload`

   If the `.reload` file doesn't exist, the `touch` or `echo` command creates it.

To Deploy an Application or Module Automatically

This task is best suited for use in a development environment.

Automatic deployment involves copying an archive file into a special autodeploy directory where the archive is automatically deployed by Eclipse GlassFish at predefined intervals. This method is useful in a development environment because it allows new code to be tested quickly. Automatic deployment is enabled by default, and is available only on the default server instance.

1. Use the `set` subcommand to adjust the autodeployment interval.

   This sets the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is `2`.

2. Use the `set` subcommand to enable JSP precompilation.

3. Copy your archive file to the autodeploy directory.

   The default location is domain-dir/autodeploy. The application will be deployed at the next interval.

   To undeploy an automatically deployed application or module, remove its archive file from the autodeploy directory.

   Deployment directories might change between Eclipse GlassFish releases.

Example 2-16 Setting the Autodeployment Interval

This example sets the autodeployment inverval to 3 seconds (default is 2).

```
asadmin> set server.admin-service.das-config.autodeploy-polling-interval-in-seconds=3
Command set executed successfully.
```

Example 2-17 Setting JSP Precompilation

This example enables JSP precompilation (default is false).

```
asadmin>
set server.admin-service.das-config.autodeploy-jsp-precompilation-enabled=true
Command set executed successfully.
```

See Also

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

To Deploy an Application or Module by Using a Deployment Plan

A Deployment Plan is a ZIP or JAR archive with a collection of descriptor files that are merged into the application or module archive being deployed. This allows deploying the same archive with different descriptors tailored for the specific environment.

Deployment Plan file for a WAR

In the deployment plan for a WAR file, the `glassfish-web.xml` file is located at the root. Files `web.xml`, `glassfish-web.xml`, `sun-web.xml`, `beans.xml`, and `faces-config.xml` will be copied into the `WEB-INF` directory of the WAR. All other descriptors (e.g. `persistence.xml`), will be copied into `WEB-INF/classes/META-INF` in the WAR.

The Deployment Plan can also contain any arbitrary files (they can also be in subdirectories), which will be unpacked into the '`WEB-INF/classes/META-INF' directory of the WAR. Any file in the Deployment Plan will overwrite the file with the same name and path in the WAR.

Deployment Plan file for an EAR

In the deployment plan for an EAR file, the `glassfish-application.xml` file is located at the root. The deployment descriptor for each module is stored according to this syntax: module-name.gf-dd-name, where the gf-dd-name depends on the module type. If a module named `MyModule` contains a CMP mappings file, the file is named `MyModule.sun-cmp-mappings.xml`. A `.dbschema` file is stored at the root level. Each `/` (forward slash) is replaced by a `#` (pound sign). All the descriptors will be unpacked into the `META-INF` directory of the EAR.

The Deployment Plan can also contain any arbitrary files (they can also be in subdirectories), which will be unpacked into the 'META-INF' directory of the EAR. Any file in the Deployment Plan will overwrite the file with the same name and path in the EAR.

Use the Deployment Plan

1. Ensure that the server is running.

   Remote commands require a running server.

2. Deploy the application or module by using the `deploy` subcommand with the `--deploymentplan` option.

Deployment directories might change between Eclipse GlassFish releases.

Example 2-18 Deploying a WAR Using a Deployment Plan

This example deploys the application in the `myrostapp.war` file according to the plan specified by the `mydeployplan-war.jar` file.

```
asadmin>deploy --deploymentplan mydeployplan-war.jar myrostapp.war
Application deployed successfully with name myrostapp.
Command deploy executed successfully.
```

Example 2-19 Deployment Plan Structure for an Web Application

This listing shows the structure of the deployment plan JAR file for a WAR file.

```
$ jar -tvf mydeployplan-war.jar
420 Thu Mar 13 15:37:48 PST 2024 glassfish-web.xml
370 Thu Mar 13 15:37:48 PST 2024 web.xml
280 Thu Mar 13 15:37:48 PST 2024 faces-config.xml
350 Thu Mar 13 15:37:48 PST 2024 beans.xml
418 Thu Mar 13 15:37:48 PST 2024 persistence.xml
```

Example 2-20 Deploying an EAR Using a Deployment Plan

This example deploys the application in the `myrostapp.ear` file according to the plan specified by the `mydeployplan.jar` file.

```
asadmin>deploy --deploymentplan mydeployplan.jar myrostapp.ear
Application deployed successfully with name myrostapp.
Command deploy executed successfully.
```

Example 2-21 Deployment Plan Structure for an Enterprise Application

This listing shows the structure of the deployment plan JAR file for an EAR file.

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 glassfish-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.glassfish-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.glassfish-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.glassfish-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.glassfish-ejb-jar.xml
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

Example 2-22 Deployment Plan Structure for an EJB Module

In the deployment plan for an EJB module, the deployment descriptor that is specific to Eclipse GlassFish is at the root level. If a standalone EJB module contains a CMP bean, the deployment plan includes the `sun-cmp-mappings.xml` and `.dbschema` files at the root level. In the following listing, the deployment plan describes a CMP bean:

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 glassfish-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

See Also

The deployment plan is part of the implementation of JSR 88. For more information about JSR 88, see the JSR 88 page at `http://jcp.org/en/jsr/detail?id=88`.

To Deploy an Application or Module in a Directory Format

This task is best suited for use in a development environment.

An expanded directory, also known as an exploded directory, contains an unassembled (unpackaged) application or module. To deploy a directory format instead of an archive, file, use the `asadmin deploy` subcommand in remote mode and specify a path to a directory instead of to an archive file. The contents of the directory must be the same as the contents of a corresponding archive file, with one exception. An application archive file contains archive files for its modules, for example `myUI.war` and `myEJB.jar`. The expanded application directory contains expanded directories for the modules, for example `myUI_war` and `myEJB_jar`, instead. .

You can change deployment descriptor files directly in the expanded directory.

If your environment is configured to use dynamic reloading, you can also dynamically reload applications or modules that are deployed from the directory. For instructions, see To Reload Changes to Applications or Modules Dynamically.

Unlike archive file deployment, directory deployment does not copy the directory contents to the remote hosts. This means that for deployment to a cluster, the directory path may exist for both the DAS and the remote server instances but may not actually correspond to the same physical location. If any target server instance cannot see the deployed directory, or finds that it contains different files from those detected by the DAS, deployment fails.

Integrated development environments (IDEs) typically use directory deployment, so you do not need to deal directly with the expanded format.

Before You Begin

On each cluster or stand-alone server instance to which the application or module is deployed, the directory must be accessible and must contain the same files as found by the DAS.

On Windows, if you are deploying a directory on a mapped drive, you must be running Eclipse GlassFish as the same user to which the mapped drive is assigned. This enables Eclipse GlassFish to access the directory.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Verify that the expanded directory contents match the archive file.

   For information about the required directory contents, see the appropriate specifications.

3. Deploy the directory by using the `deploy` subcommand and specifying the path to the expanded directory.

Deployment directories might change between Eclipse GlassFish releases.

Example 2-23 Deploying an Application From a Directory

This example deploys the expanded directory `/apps/MyApp` for the `hello` application.

```
asadmin> deploy --name hello /apps/MyApp
Application deployed successfully with name hello.
Command deploy executed successfully.
```

See Also

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

Modifying the Configuration of a Web Application or Module

You can modify the configuration of a web application or a module by modifying the deployment descriptors and then repackaging and redeploying the application.

The instructions in this section enable you to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application. If the application or module entry is unset, its value reverts to the value, if any, that is set in the application's deployment descriptor.

The following topics are addressed here:

To Set a Web Context Parameter

Use the `set-web-context-param` subcommand in remote mode to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application. By using this subcommand, you are either adding a new parameter that did not appear in the original web module's descriptor, or overriding the descriptor's setting of the parameter.

If the `--ignoreDescriptorItem` option is set to `true`, then the server ignores any setting for that context parameter in the descriptor, which means you do not need to specify an overriding value on the `set-web-context-param` subcommand. The server behaves as if the descriptor had never contained a setting for that context parameter.

This subcommand sets a servlet context-initialization parameter of one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Jakarta EE) application

Before You Begin

The application must already be deployed. Otherwise, an error occurs.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Set a servlet context-initialization parameter by using the `set-web-context-param` subcommand.

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

Example 2-24 Setting a Servlet Context-Initialization Parameter for a Web Application

This example sets the servlet context-initialization parameter javax.faces.STATE_SAVING_METHOD of the web application `basic-ezcomp` to client.

```
asadmin> set-web-context-param --name=javax.faces.STATE_SAVING_METHOD
--description="The location where the application?s state is preserved"
--value=client basic-ezcomp
Command set-web-context-param executed successfully.
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help set-web-context-param` at the command line.

To Unset a Web Context Parameter

Use the `unset-web-context-param` subcommand in remote mode to unset an environment entry for a deployed web application or module that has been set by using the `set-web-env-entry` subcommand. There is no need to modify the application's deployment descriptors and repackage and redeploy the application.

This subcommand unsets an environment entry for one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Jakarta EE) application

When an entry is unset, its value reverts to the value, if any, that is set in the application's deployment descriptor. This subcommand cannot be used to change the value of an environment entry that is set in an application's deployment descriptor. Instead, use the `set-web-context-param` subcommand for this purpose.

Before You Begin

The application must already be deployed, and the entry must have previously been set by using the `set-web-env-entry` subcommand. Otherwise, an error occurs.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Unset an environment entry by using the `unset-web-context-param` subcommand.

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

Example 2-25 Unsetting a Servlet Context-Initialization Parameter for a Web Application

This example unsets the servlet context-initialization parameter

javax.faces.STATE_SAVING_METHOD of the web application `basic-ezcomp`.

```
asadmin> unset-web-context-param
--name=javax.faces.STATE_SAVING_METHOD basic-ezcomp
Command unset-web-context-param executed successfully.
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help unset-web-context-param` at the command line.

To List Web Context Parameters

Use the `list-web-context-param` subcommand in remote mode to list the parameters that have previously been set by using the `set-web-context-param` subcommand. The subcommand does not list parameters that are set only in the application's deployment descriptor. For each parameter, the following information is displayed:

- The name of the parameter
- The value to which the parameter is set
- The value of the `--ignoreDescriptorItem` option of the `set-web-context-param` subcommand that was specified when the parameter was set
- The description of the parameter or `null` if no description was specified when the parameter was set

1. Ensure that the server is running.

   Remote commands require a running server.

2. List servlet context-initialization parameters by using the `list-web-context-param` subcommand.

Example 2-26 Listing Servlet Context-Initialization Parameters for a Web Application

This example lists all servlet context-initialization parameters of the web application `basic-ezcomp` that have been set by using the `set-web-context-param` subcommand. Because no description was specified when the javax.faces.PROJECT_STAGE parameter was set, null is displayed instead of a description for this parameter.

```
asadmin> list-web-context-param basic-ezcomp
javax.faces.STATE_SAVING_METHOD = client ignoreDescriptorItem=false
//The location where the application's state is preserved
javax.faces.PROJECT_STAGE = null ignoreDescriptorItem=true //null
Command list-web-context-param executed successfully.
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help list-web-`

`context-param` at the command line.

To Set a Web Environment Entry

An application uses the values of environment entries to customize its behavior or presentation. Use the `set-web-env-entry` subcommand in remote mode to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application. By using this subcommand, you are either adding a new parameter that did not appear in the original web module's descriptor, or overriding the descriptor's setting of the parameter.

If you the `--ignoreDescriptorItem` option is set to `true`, then the server ignores any setting for that environment entry in the descriptor, which means you do not need to specify an overriding value on the `set-web-env-entry` subcommand. The server behaves as if the descriptor had never contained a setting for that environment entry.

This subcommand sets an environment entry for one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Jakarta EE) application

Before You Begin

The application must already be deployed. Otherwise, an error occurs.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Set an environment entry for a deployed web application or module by using the `set-web-env-entry` subcommand.

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

Example 2-27 Setting an Environment Entry for a Web Application

This example sets the environment entry `Hello User` of the application hello to `techscribe`. The Java type of this entry is `java.lang.String`.

```
asadmin> set-web-env-entry --name="Hello User"
--type=java.lang.String --value=techscribe
--description="User authentication for Hello appplication" hello
Command set-web-env-entry executed successfully
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help set-web-env-entry` at the command line.

To Unset a Web Environment Entry

Use the `unset-web-env-entry` subcommand in remote mode to unset an environment entry for a deployed web application or module.

1. Ensure that the server is running.

   Remote commands require a running server.

2. Unset a web environment entry by using the `unset-web-env-entry` subcommand.

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

Example 2-28 Unsetting an Environment Entry for a Web Application

This example unsets the environment entry `Hello User` of the web application `hello`.

```
asadmin> unset-web-env-entry --name="Hello User" hello
Command unset-web-env-entry executed successfully.
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help unset-web-env-entry` at the command line.

To List Web Environment Entries

Use the `list-web-env-entry` subcommand to list environment entries for a deployed web application or module. For each entry, the following information is displayed:

- The name of the entry
- The Java type of the entry
- The value to which the entry is set
- The description of the entry or null if no description was specified when the entry was set
- The value of the `--ignoreDescriptorItem` option of the `set-web-env-entry` subcommand that was specified when the entry was set

1. Ensure that the server is running.

   Remote commands require a running server.

2. List the environment entries by using the `list-web-env-entry` subcommand.

Example 2-29 Listing Environment Entries for a Web Application

This example lists all environment entries that have been set for the web application `hello` by using the `set-web-env-entry` subcommand.

```
asadmin> list-web-env-entry hello
```

```
Hello User (java.lang.String) = techscribe ignoreDescriptorItem=false
//User authentication for Hello appplication
Hello Port (java.lang.Integer) = null ignoreDescriptorItem=true //null
Command list-web-env-entry executed successfully.
```

See Also

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

Web Module Deployment Guidelines

The following guidelines apply to deploying a web module in Eclipse GlassFish:

- Context Root. When you deploy a web module, if you do not specify a context root, the default is the name of the WAR file without the `.war` extension. The web module context root must be unique within the server instance.

  The domain administration server (DAS) in Eclipse GlassFish versions 2.1.1 and later supports the deployment of multiple web applications using the same web context root as long as those applications are deployed to different Eclipse GlassFish stand-alone instances. Deploying multiple applications using the same context root within a single instance produces an error.

- Data Source. If a web application accesses a `DataSource` that is not specified in a `resource-ref` in `glassfish-web.xml`, or there is no `glassfish-web.xml` file, the `resource-ref-name` defined in `web.xml` is used. A warning message is logged, recording the JNDI name that was used to look up the resource.
- Virtual Servers. If you deploy a web application and do not specify any assigned virtual servers, the web application is assigned to all currently-defined virtual servers with the exception of the virtual server with ID `__asadmin`, which is reserved for administrative purposes. If you then create additional virtual servers and want to assign existing web applications to them, you must redeploy the web applications.
- HTTP Sessions. If a web application is undeployed, all its HTTP sessions will be invalidated and removed, unless the application is being undeployed as part of a redeployment and the `--keepstate` deployment option was set to true. This option is not supported and ignored in a clustered environment. See Example 2-8.

  For information about HTTP session persistence, see the Eclipse GlassFish High Availability Administration Guide.

- Load Balancing. See the Eclipse GlassFish High Availability Administration Guide for information about load balancing.
- JSP Precompilation. You can precompile JSP files during deployment by checking the appropriate box in the Administration Console, or by using the `--precompilejsp` option of the `deploy` subcommand.

  You can keep the generated source for JSP files by adding the `keepgenerated` flag to the `jsp-config` element in `glassfish-web.xml`. For example:

```
<glassfish-web-app>
   ...
   <jsp-config>
      <property name=keepgenerated value=true />
   </jsp-config>
</glassfish-web-app>
```

If you include this property when you deploy the WAR file, the generated source is kept in domain-dir/generated/jsp/app-name/module-name for an application, or domain-dir/generated/jsp/module-name for an individually-deployed web module.

For more information about JSP precompilation, see jsp-config.

- Web Context Parameters. You can set web context parameters after deployment. See the following sections:
  - To Set a Web Context Parameter
  - To Unset a Web Context Parameter
  - To List Web Context Parameters
- Web Environment Entries. You can set web environment entries after deployment. See the following sections:
  - To Set a Web Environment Entry
  - To Unset a Web Environment Entry
  - To List Web Environment Entries

EJB Module Deployment Guidelines

The Eclipse GlassFish Web Profile supports the EJB 3.1 Lite specification, which allows enterprise beans within web applications, among other features. The Eclipse GlassFish Full Platform Profile supports the entire EJB 3.1 specification. For details, see JSR 318

The following guidelines apply to deploying an EJB module in Eclipse GlassFish:

- JNDI Name. — If no JNDI name for the EJB JAR module is specified in the jndi-name element immediately under the ejb element in glassfish-ejb-jar.xml, or there is no glassfish-ejb-jar.xml file, a default, non-clashing JNDI name is derived. A warning message is logged, recording the JNDI name used to look up the EJB JAR module.

  Because the EJB 3.1 specification defines portable EJB JNDI names, there is less need for Eclipse GlassFish specific JNDI names. By default, Eclipse GlassFish specific default JNDI names are applied automatically for backward compatibility. To disable Eclipse GlassFish specific JNDI names for an EJB module, set the value of the <disable-nonportable-jndi-names> element in the glassfish-ejb-jar.xml file to true. The default is false.

- Stateful Session Bean and Timer State. — Use the --keepstate option of the redeploy

subcommand or the `<keepstate>` element in the `glassfish-ejb-jar.xml` file to retain stateful session bean instances and persistently created EJB timers across redeployments. The `--keepstate` option of the `redeploy` subcommand takes precedence. The default for both is `false`. This option is not supported and ignored in a clustered environment.

Some changes to an application between redeployments can prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class. Other examples would be changes to EJB names, or adding or removing EJBs to or from an application.

- EJB Singletons. — EJB Singletons are created for each server instance in a cluster, and not once per cluster.
- Stubs and Ties. — Use the `get-client-stubs` subcommand in remote mode to retrieve stubs and ties.
- Compatibility of JAR Visibility Requirements. — Use the `compatibility` element of the `glassfish-application.xml` or `glassfish-ejb-jar.xml` file to specify the Eclipse GlassFish release with which to be backward compatible in terms of JAR visibility requirements for applications. The current allowed value is `v2`, which refers to Eclipse GlassFish version 2 or Eclipse GlassFish version 9.1 or 9.1.1. Starting in Jakarta EE 6, the Jakarta EE specification imposes stricter requirements than Jakarta EE 5 did on which JAR files can be visible to various modules within an EAR file. Setting this element to `v2` removes these Jakarta EE 6 and later restrictions.

Deploying a Connector Module

Deploying a stand-alone connector module allows multiple deployed Java EE applications to share the connector module. A resource adapter configuration is automatically created for the connector module.

The following topics are addressed here:

- To Deploy and Configure a Stand-Alone Connector Module
- Redeploying a Stand-Alone Connector Module
- Deploying and Configuring an Embedded Resource Adapter

To Deploy and Configure a Stand-Alone Connector Module

As an alternative to Step 3 through Step 6, you can define application-scoped resources in the `glassfish-resources.xml` deployment descriptor. For more information, see Application-Scoped Resources.

1. Ensure that the server is running. Remote commands require a running server.
2. Deploy the connector module by using the `deploy` subcommand.
3. Configure connector connection pools for the deployed connector module.

   Use the `create-connector-connection-pool` subcommand. For procedures, see "To Create a Connector Connection Pool" in Eclipse GlassFish Administration Guide.

4. Configure connector resources for the connector connection pools.

   Use the `create-resource-adapter-config` subcommand. For procedures, see "To Create Configuration Information for a Resource Adapter" in Eclipse GlassFish Administration Guide. If needed, you can override the default configuration properties of a resource adapter.

   This step associates a connector resource with a JNDI name.

5. Configure a resource adapter.

   Use the `create-resource-adapter-config` subcommand. For procedures, see "To Create Configuration Information for a Resource Adapter" in Eclipse GlassFish Administration Guide. If needed, you can override the default configuration properties of a resource adapter.

6. If needed, create an administered object for an inbound resource adapter.

   Use the `create-admin-object` subcommand. For procedures, see "To Create an Administered Object" in Eclipse GlassFish Administration Guide.

Redeploying a Stand-Alone Connector Module

Redeployment of a connector module maintains all connector connection pools, connector resources, and administered objects defined for the previously deployed connector module. You do not need to reconfigure any of these resources.

However, you should redeploy any dependent modules. A dependent module uses or refers to a connector resource of the redeployed connector module. Redeployment of a connector module results in the shared class loader reloading the new classes. Other modules that refer to the old resource adapter classes must be redeployed to gain access to the new classes. For more information about class loaders, see "Class Loaders" in Eclipse GlassFish Application Development Guide.

During connector module redeployment, the server log provides a warning indicating that all dependent applications should be redeployed. Client applications or application components using the connector module's resources may throw class cast exceptions if dependent applications are not redeployed after connector module redeployment.

To disable automatic redeployment, set the `--force` option to `false`. In this case, if the connector module has already been deployed, Eclipse GlassFish provides an error message.

Deploying and Configuring an Embedded Resource Adapter

A connector module can be deployed as a Jakarta EE component in a Jakarta EE application. Such connectors are only visible to components residing in the same Jakarta EE application. Deploy this application as you would any other Jakarta EE application.

You can create new connector connection pools and connector resources for a connector module embedded within a Jakarta EE application by prefixing the connector name with app-name`#. For example, if an application `appX.ear` has `jdbcra.rar` embedded within it, the connector connection pools and connector resources refer to the connector module as `appX#jdbcra`.

An embedded connector module cannot be undeployed using the name app-name`#`connector-name. To undeploy the connector module, you must undeploy the application in which it is embedded.

The association between the physical JNDI name for the connector module in Eclipse GlassFish and the logical JNDI name used in the application component is specified in the Eclipse GlassFish-specific XML descriptor `glassfish-ejb-jar.xml`.

Assembling and Deploying an Application Client Module

Deployment is necessary for application clients that communicate with EJB components or that use Java Web Start launch support. Java Web Start is supported for application clients and for applications that contain application clients. By default, Java Web Start is enabled in application clients and in Eclipse GlassFish.

The Application Client Container is supported only in the Eclipse GlassFish Full Platform Profile, not in the Web Profile.

The following topics are addressed here:

- To Assemble and Deploy an Application Client
- To Prepare Another Machine for Running an Application Client
- To Undeploy an Application Client

To Assemble and Deploy an Application Client

1. Assemble the necessary client components.

   The client JAR file is created.

2. Assemble the EJB components that are to be accessed by the client.

   The EJB JAR file is created.

3. Assemble the client and EJB JAR files together in an EAR.

   An EAR file contains all the components of the application.

4. Deploy the application.

   Instructions are contained in To Deploy an Application or Module.

5. If you are using the `appclient` script to run the application client, retrieve the client files.

   The client artifacts contain the ties and necessary classes for the application client. In this release of Eclipse GlassFish, the client artifacts include multiple files. You can use either the `get-client-stubs` subcommand or the `--retrieve` option of the `deploy` subcommand, but you do not need to use both. * Use the `deploy` subcommand with the `--retrieve` option to retrieve the client

files as part of deploying the application. * Use the `get-client-stubs` subcommand to retrieve client files for a previously-deployed application.

6. Test the client on the Eclipse GlassFish machine in one of the following ways:
   - If Java Web Start is enabled for the application client, use the Launch link on the Application Client Modules.
   - Run an application client by using the `appclient` script.

     The `appclient` script is located in the as-install`/bin` directory.

     If you are using the default server instance, the only required option is `-client`, which points to the client JAR file. For example:

     ```
     appclient -client converterClient.jar
     ```

     The -xml parameter, which specifies the location of the `sun-acc.xml` file, is also required if you are not using the default instance.

See Also

For more detailed information about the `appclient` script, see `appclient`(1M).

For more detailed information about creating application clients, see "Developing Java Clients" in Eclipse GlassFish Application Development Guide. This chapter includes information on the following topics:

- Accessing EJB components and JMS resources from application clients
- Connecting to a remote EJB module through a firewall
- Using Java Web Start and creating a custom JNLP file
- Using libraries with application clients
- Specifying a splash screen, login retries, and other customizations

To Prepare Another Machine for Running an Application Client

If Java Web Start is enabled, the default URL format for an application is `http://`host:port`/``context-root. For example:

```
http://localhost:80/myapp
```

The default URL format for a standalone application client module is `http://`host:port`/`module-id. For example:

```
http://localhost:80/myclient
```

To set a different URL for an application client, set the `context-root` subelement of the `java-web-`

`start-access` element in the `glassfish-application-client.xml` file.

If the context-root or module-id is not specified during deployment, the name of the EAR or JAR file without the `.ear` or `.jar` extension is used. For an application, the relative path to the application client JAR file is also included. If the application or module is not in EAR or JAR file format, a context-root or module-id is generated. Regardless of how the context-root or module-id is determined, it is written to the server log. For details about naming, see Naming Standards.

Before You Begin

This task applies if you want to use the `appclient` script to run the application client on a system other than where the server runs.

1. Create the application client package JAR file.

   Use the `package-appclient` script in the as-install`/bin` directory. This JAR file is created in the as-install`/lib/appclient` directory.

2. Copy the application client package JAR file to the client machine.
3. Extract the contents of the JAR file.

   For example: `jar xf` filename`.jar`

4. Configure the `sun-acc.xml` file.

   If you used the `package-appclient` script, this file is located in the `appclient/appserv/lib/appclient` directory by default.

5. Configure the `asenv.conf` (`asenv.bat` on Windows) file.

   This file is located in `appclient/appserv/bin` by default if you used the `package-appclient` script.

6. Copy the client JAR file to the client machine.

   You are now ready to run the client.

See Also

For more detailed information about Java Web Start and the `package-appclient` script, see `appclient`(1M).

To Undeploy an Application Client

After application clients are downloaded, they remain on the client until they are manually removed. Use the Java Web Start control panel to discard downloaded application clients that used Java Web Start.

If you undeploy an application client, you can no longer use Java Web Start, or any other mechanism, to download that application client because it might be in an inconsistent state. If you try to launch an application client that was previously downloaded (even though the server side of the application client is no longer present), the results are unpredictable unless the application

client has been written to tolerate such situations.

You can write your application client so that it detects failures in contacting server-side components, but continues running. In this case, Java Web Start can run an undeployed application client while the client is cached locally. For example, your application client can be written to detect and then recover from `javax.naming.NamingException` when locating a resource, or from `java.rmi.RemoteException` when referring to a previously-located resource that becomes inaccessible.

Lifecycle Module Deployment Guidelines

A lifecycle module, also called a lifecycle listener module, provides a means of running long or short Java-based tasks within the Eclipse GlassFish environment, such as instantiation of singletons or RMI servers. Lifecycle modules are automatically initiated at server startup and are notified at various phases of the server life cycle. All lifecycle module interfaces are in the as-install`/modules/glassfish-api.jar` file.

For general information about lifecycle modules, see "Developing Lifecycle Listeners" in Eclipse GlassFish Application Development Guide.

You can deploy a lifecycle module using the `create-lifecycle-module` subcommand. Do not use `asadmin deploy` or related commands.

You do not need to specify a classpath for the lifecycle module if you place it in the domain-dir`/lib` or domain-dir`/lib/classes` directory for the Domain Administration Server (DAS). Do not place it in the `lib` directory for a particular server instance, or it will be deleted when that instance synchronizes with the Eclipse GlassFish.

After you deploy a lifecycle module, you must restart the server. During server initialization, the server instantiates the module and registers it as a lifecycle event listener.

If the `--failurefatal` option of `create-lifecycle-module` is set to `true` (the default is `false`), lifecycle module failure prevents server initialization or startup, but not shutdown or termination.

Web Service Deployment Guidelines

If you installed the Web Profile, web services are not supported unless the optional Metro Web Services Stack add-on component is downloaded. Without the Metro add-on component, a servlet or EJB component cannot be a web service endpoint, and the `glassfish-web.xml` and `glassfish-ejb-jar.xml` elements related to web services are ignored.

The following guidelines apply when deploying a web service in Eclipse GlassFish:

- Web Service Endpoint. Deploy a web service endpoint to Eclipse GlassFish as you would any servlet or stateless session bean. If the deployed application or module has a web service endpoint, the endpoint is detected automatically during deployment. The Eclipse GlassFish

-specific deployment descriptor files, `glassfish-web.xml` and `glassfish-ejb-jar.xml`, provide optional web service enhancements in their `webservice-endpoint` and `webservice-description` elements.

- Web Service Management. Web service management is fully supported in the Administration Console. After the application or module is deployed, click the Web Service component. The table in the right frame lists deployed web service endpoints.

For more information about web services, see "Developing Web Services" in Eclipse GlassFish Application Development Guide.

OSGi Bundle Deployment Guidelines

To deploy an OSGi bundle using the Administration Console, select Other from the Type drop-down list and check the OSGI Type checkbox.

To deploy an OSGi bundle using the `asadmin deploy` command, set the `--type` option to the value `osgi`. For example:

```
asadmin> deploy --type=osgi MyBundle.jar
```

To automatically deploy an OSGi bundle, copy the bundle archive to the domain-dir`/autodeploy/bundles` directory.

For components packaged as OSGi bundles (`--type=osgi`), the `deploy` subcommand accepts properties arguments to wrap a WAR file as a WAB (Web Application Bundle) at the time of deployment. The subcommand looks for a key named `UriScheme` and, if present, uses the key as a URL stream handler to decorate the input stream. Other properties are used in the decoration process. For example, the Eclipse GlassFish OSGi web container registers a URL stream handler named `webbundle`, which is used to wrap a plain WAR file as a WAB.

Transparent JDBC Connection Pool Reconfiguration

In this Eclipse GlassFish release, reconfiguration of a JDBC connection pool due to attribute or property changes can be transparent to the applications or modules that use the pool, even if pool reconfiguration results in pool recreation. You do not need to redeploy the application or module.

To enable transparent pool reconfiguration, set the `dynamic-reconfiguration-wait-timeout-in-seconds` property. This property specifies the timeout for dynamic reconfiguration of the pool. In-progress connection requests must complete before this timeout expires or they must be retried. New connection requests wait for this timeout to expire before acquiring connections to the reconfigured pool. If this property exists and has a positive value, it is enabled.

You can set this property in the `glassfish-resources.xml` file. For more information, see the property descriptions under `jdbc-connection-pool`.

For JDBC connection pools that are not application-scoped, use the `set` subcommand to set this

property. For example, to configure `mypool` on `myserver`, type the following all on one line:

```
asadmin> set myserver.resources.jdbc-connection-pool.mypool.property.
dynamic-reconfiguration-wait-timeout-in-seconds=30
```

Application-Scoped Resources

You can define an application-scoped JDBC resource or other resource for an enterprise application, web module, EJB module, connector module, or application client module. This allows single-step deployment for resource-dependent modules and applications. An application-scoped resource has the following characteristics:

- It is available only to the module or application that defines it.
- It cannot be referenced or looked up by other modules or applications.
- It is created during deployment, destroyed during undeployment, and recreated during redeployment.
- It is free from unexpected resource starvation or delay in acquiring connections because no other application or module competes for accesses to it.

The following resource types can be application-scoped:

- JDBC connection pools
- JDBC resources
- Connector connection pools
- Connector resources
- Resource adapters
- External JNDI resources
- Custom resources
- Admin object resources
- Jakarta Mail resources

Deployment Descriptor. An application-scoped resource is defined in the `glassfish-resources.xml` deployment descriptor file. This file is placed in the `META-INF` directory of the module or application archive. For web applications or modules, this file is placed in the `WEB-INF` directory. If any submodule archives of an enterprise application archive have their own `glassfish-resources.xml` files, the resource definitions are scoped to those modules only. For more information about the `glassfish-resources.xml` file, see Eclipse GlassFish Deployment Descriptor Files and Elements of the Eclipse GlassFish Deployment Descriptors.

Naming. Application-scoped resource JNDI names begin with `java:app` or `java:module`. If one of these prefixes is not specified in the JNDI name, it is added. For example, application-scoped databases have JNDI names in the following format: `java:app/jdbc/`DataSourceName or `java:module/jdbc/`DataSourceName. This is in accordance with the naming scopes introduced in

the Jakarta EE 6 Specification.

Errors. Application-scoped resource definitions with same resource name, resource type, attributes, and properties are duplicates. These generate `WARNING` level log messages and deployment continues. Definitions with the same resource name and type but different attributes or properties are conflicts and cause deployment failure. When an application or module tries to look up a scoped resource that does not belong to it, a naming exception is thrown.

Redeployment. When an application or module is undeployed, its scoped resources are deleted. During redeployment, resources are destroyed and recreated based on changes in the `glassfish-resources.xml` file. To preserve old resource definitions during redeployment, use the `preserveAppScopedResources` property of the `redeploy` (or `deploy --force=true`) subcommand. For example:

```
asadmin> redeploy --property preserveAppScopedResources=true MyApp.ear

asadmin> deploy --force=true --property preserveAppScopedResources=true MyApp.ear
```

For more information, see `redeploy`(1) and `deploy`(1).

Listing. Use the `--resources` option of the `list-applications` subcommand to list application-scoped resources. Use the `--subcomponents` option in addition to list scoped resources for enterprise application modules or for module subcomponents. To list scoped resources for subcomponents only, use the `--resources` option of the `list-subcomponents` subcommand

For more information, see `list-applications`(1) and `list-sub-components`(1).

Restrictions. Use of application-scoped resources is subject to the following restrictions:

- `resource-adapter-config` and `connector-work-security-map` — These can only be specified in the `glassfish-resources.xml` file of the corresponding connector module. In an enterprise application, the `resource-adapter-config` or `connector-work-security-map` for an embedded connector module must be specified in the `glassfish-resources.xml` file of the connector module. You cannot specify a `resource-adapter-config` or `connector-work-security-map` in an application for a connector module that is not part of the application.
- Resource to connection pool cross references — A module-level `jdbc-resource` cannot reference an application-level `jdbc-connection-pool`. Likewise, a module-level `connector-resource` cannot reference an application-level `connector-connection-pool`.
- Global resources — Defining `java:global` JNDI names is not supported.
- Cross definitions — Defining `java:app` JNDI names at the module level is not supported.

A The `asadmin` Deployment Subcommands

This appendix lists the `asadmin` deployment subcommands that are included with this release of the Eclipse GlassFish software. For information on additional `asadmin` subcommands, see "Subcommands for the `asadmin` Utility" in Eclipse GlassFish Administration Guide or see the Eclipse GlassFish Reference Manual.

`add-library`

Adds one or more library JAR files to Eclipse GlassFish. You can specify whether the libraries are added to the Common class loader directory, the Java optional package directory, or the application-specific class loader directory.

`create-application-ref`

Creates a reference from a cluster or an unclustered server instance to a previously deployed Jakarta EE application or module. This effectively results in the application element being deployed and made available on the targeted instance or cluster.

`create-lifecycle-module`

Creates a lifecycle module. A lifecycle module provides a means of running a short or long duration Java-based task at a specific stage in the server life cycle.

`delete-application-ref`

Removes a reference from a cluster or an unclustered server instance to a previously deployed Jakarta EE application or module. This effectively results in the application element being undeployed on the targeted instance or cluster.

`delete-lifecycle-module`

Deletes a lifecycle module.

`deploy`

Deploys an enterprise application, web application, EJB module, connector module, or application client module. If the component is already deployed or already exists, you can forcefully redeploy if you set the `--force` option to `true`. A directory can also be deployed. Supported in remote mode only. For usage instructions, see To Deploy an Application or Module.

`deploydir`

This subcommand is deprecated. Use the `deploy` subcommand instead.

`disable`

Immediately deactivates the named application or module. If the component has not been deployed, an error message is returned. Supported in remote mode only. For usage instructions, seeTo Disable an Application or Module.

`enable`

Enables the specified application or module. If the component has not been deployed, an error message is returned. If the component is already enabled, then it is re-enabled. Supported in remote mode only. For usage instructions, see To Enable an Application or Module.

`get-client-stubs`

Gets the client stubs JAR file for an application client module or an application containing the application client module, from the server machine to the local directory. For usage instructions, see EJB Module Deployment Guidelines.

`list-applications`

Lists deployed Jakarta EE applications and modules. Optionally lists subcomponents and scoped resources. If the `--type` option is not specified, all applications and modules are listed. Supported in remote mode only. For usage instructions, see To List Deployed Applications or Modules.

`list-application-refs`

Lists Jakarta EE applications and modules deployed on the specified target server instance or cluster.

`list-libraries`

Lists library JAR files that have been added to Eclipse GlassFish. You can specify whether to list libraries in the Common class loader directory, the Java optional package directory, or the application-specific class loader directory.

`list-lifecycle-modules`

Lists lifecycle modules.

`list-components`

This subcommand is deprecated. Use the `list-applications` subcommand instead.

`list-sub-components`

Lists EJBs or servlets in a deployed module or in a module of the deployed application. If a module is not identified, all modules are listed. To display a specific module in an application, you must specify the module name and the `--appname` option. Supported in remote mode only. For usage instructions, see To List Deployed Applications or Modules.

`list-web-context-param`

Lists servlet context-initialization parameters of a deployed web application or module. Supported in remote mode only. For usage instructions, see To List Web Context Parameters.

`list-web-env-entry`

Lists environment entries for a deployed web application or module. Supported in remote mode only. For usage instructions, see To List Web Environment Entries.

`redeploy`

Overwrites an application or module that is already deployed. Supported in remote mode only. For usage instructions, see To Redeploy an Application or Module.

`remove-library`

Removes one or more library JAR files from Eclipse GlassFish. You can specify whether the libraries are removed from the Common class loader directory, the Java optional package directory, or the application-specific class loader directory.

`set-web-context-param`

Sets a servlet context-initialization parameter of a deployed web application or module. Supported in remote mode only. For usage instructions, see To Set a Web Context Parameter.

`set-web-env-entry`

Sets an environment entry for a deployed web application or module. Supported in remote mode only. For usage instructions, see To Set a Web Environment Entry.

`show-component-status`

Shows the status of a deployed component. The possible statuses include `enabled` or `disabled`. Supported in remote mode only. For usage instructions, see To List Deployed Applications or Modules.

`undeploy`

Uninstalls the specified deployed application or module. Supported in remote mode only. For usage instructions, see To Undeploy an Application or Module.

`unset-web-context-param`

Unsets a servlet context-initialization parameter of a deployed web application or module. Supported in remote mode only. For usage instructions, see To Unset a Web Context Parameter.

`unset-web-env-entry`

Unsets an environment entry for a deployed web application or module. Supported in remote mode only. For usage instructions, see To Unset a Web Environment Entry.

B Eclipse GlassFish Deployment Descriptor Files

This appendix describes the element hierarchies in the Eclipse GlassFish deployment descriptors that are included in this release of the Eclipse GlassFish software.

The following topics are addressed here:

About the Eclipse GlassFish Deployment Descriptors

Each deployment descriptor XML file has a corresponding Document Type Definition (DTD) file, which defines the elements, data, and attributes that the deployment descriptor file can contain. For example, the `glassfish-application_6_0-1.dtd` file defines the structure of the `glassfish-application.xml` file. The DTD files for the Eclipse GlassFish deployment descriptors are located in the as-install`/lib/dtds` directory.

The Eclipse GlassFish deployment descriptor files must be readable and writable by the file owners. In each deployment descriptor file, subelements must be defined in the order in which they are listed under each Subelements heading, unless otherwise noted. For general information about DTD files and XML, see the XML specification at `http://www.w3.org/TR/REC-xml`.

Do not edit the DTD files; their contents change only with new versions of Eclipse GlassFish.

The following table lists the Eclipse GlassFish deployment descriptors and their DTD files.

Table B-1 Eclipse GlassFish Deployment Descriptors and DTDs

Deployment Descriptor	DTD File	Description
`glassfish-application.xml`	`glassfish-application_6_0-1.dtd`	Configures an entire Jakarta EE application (EAR file).
`glassfish-web.xml`	`glassfish-web-app_3_0-1.dtd`	Configures a web application (WAR file).

Deployment Descriptor	DTD File	Description
glassfish-ejb-jar.xml	glassfish-ejb-jar_3_1-1.dtd	Configures an enterprise bean (EJB JAR file).
sun-cmp-mappings.xml	sun-cmp-mapping_1_2.dtd	Configures container-managed persistence for an EJB 2.0 or 2.1 entity bean.
glassfish-application-client.xml	glassfish-application-client_6_0-1.dtd	Configures an Application Client Container (ACC) client (JAR file).
sun-acc.xml	sun-application-client-container_1_2.dtd	Configures the Application Client Container. This is more of a configuration file than a deployment descriptor. Eclipse GlassFish provides a default file in the domain-dir/config directory. Specifying a different file is optional.
glassfish-resources.xml	glassfish-resources_1_5.dtd	Configures application-scoped resources.

The sun-application.xml, sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml, and sun-resources.xml deployment descriptors are supported for backward compatibility.

The glassfish-application.xml File

The glassfish-application.xml file configures an entire Jakarta EE application (EAR file). The element hierarchy is as follows:

```
glassfish-application
.  web
.  .  web-uri
.  .  context-root
.  pass-by-reference
.  unique-id
.  security-role-mapping
.  .  role-name
.  .  principal-name
.  .  group-name
.  realm
.  ejb-ref
.  .  ejb-ref-name
```

```
.  .  jndi-name
.  resource-ref
.  .  res-ref-name
.  .  jndi-name
.  .  default-resource-principal
.  .  .  name
.  .  .  password
.  resource-env-ref
.  .  resource-env-ref-name
.  .  jndi-name
.  service-ref
.  .  service-ref-name
.  .  port-info
.  .  .  service-endpoint-interface
.  .  .  wsdl-port
.  .  .  .  namespaceURI
.  .  .  .  localpart
.  .  .  stub-property
.  .  .  .  name
.  .  .  .  value
.  .  .  call-property
.  .  .  .  name
.  .  .  .  value
.  .  .  message-security-binding
.  .  .  .  message-security
.  .  .  .  .  message
.  .  .  .  .  .  java-method
.  .  .  .  .  .  .  method-name
.  .  .  .  .  .  .  method-params
.  .  .  .  .  .  .  .  method-param
.  .  .  .  .  .  operation-name
.  .  .  .  .  request-protection
.  .  .  .  .  response-protection
.  .  call-property
.  .  .  name
.  .  .  value
.  .  wsdl-override
.  .  service-impl-class
.  .  service-qname
.  .  .  namespaceURI
.  .  .  localpart
.  message-destination-ref
.  .  message-destination-ref-name
.  .  jndi-name
.  message-destination
.  .  message-destination-name
.  .  jndi-name
.  archive-name
.  compatibility
.  keep-state
```

```
.  version-identifier
```

Here is a sample `glassfish-application.xml` file:

```
<!DOCTYPE glassfish-application PUBLIC "-//GlassFish.org//DTD
GlassFish Application Server 3.1 Jakarta EE Application 6.0//EN"
"http://glassfish.org/dtds/glassfish-application_6_0-1.dtd">
<glassfish-application>
   <unique-id>67488732739338240</unique-id>
</glassfish-application>
```

The glassfish-web.xml File

The `glassfish-web.xml` file configures a web application (WAR file). The element hierarchy is as follows:

```
glassfish-web-app
.  context-root
.  security-role-mapping
.  .  role-name
.  .  principal-name
.  .  group-name
.  servlet
.  .  servlet-name
.  .  principal-name
.  .  webservice-endpoint
.  .  .  port-component-name
.  .  .  endpoint-address-uri
.  .  .  login-config
.  .  .  .  auth-method
.  .  .  message-security-binding
.  .  .  .  message-security
.  .  .  .  .  message
.  .  .  .  .  .  java-method
.  .  .  .  .  .  .  method-name
.  .  .  .  .  .  .  method-params
.  .  .  .  .  .  .  .  method-param
.  .  .  .  .  .  operation-name
.  .  .  .  .  request-protection
.  .  .  .  .  response-protection
.  .  .  transport-guarantee
.  .  .  service-qname
.  .  .  tie-class
.  .  .  servlet-impl-class
.  .  .  debugging-enabled
.  .  .  property (with attributes)
```

```
.  .  .  .  description
.  idempotent-url-pattern
.  session-config
.  .  session-manager
.  .  .  manager-properties
.  .  .  .  property (with attributes)
.  .  .  .  .  description
.  .  .  store-properties
.  .  .  .  property (with attributes)
.  .  .  .  .  description
.  .  session-properties
.  .  .  property (with attributes)
.  .  .  .  description
.  .  cookie-properties
.  .  .  property (with attributes)
.  .  .  .  description
.  ejb-ref
.  .  ejb-ref-name
.  .  jndi-name
.  resource-ref
.  .  res-ref-name
.  .  jndi-name
.  .  default-resource-principal
.  .  .  name
.  .  .  password
.  resource-env-ref
.  .  resource-env-ref-name
.  .  jndi-name
.  service-ref
.  .  service-ref-name
.  .  port-info
.  .  .  service-endpoint-interface
.  .  .  wsdl-port
.  .  .  .  namespaceURI
.  .  .  .  localpart
.  .  .  stub-property
.  .  .  .  name
.  .  .  .  value
.  .  .  call-property
.  .  .  .  name
.  .  .  .  value
.  .  .  message-security-binding
.  .  .  .  message-security
.  .  .  .  .  message
.  .  .  .  .  .  java-method
.  .  .  .  .  .  .  method-name
.  .  .  .  .  .  .  method-params
.  .  .  .  .  .  .  .  method-param
.  .  .  .  .  .  operation-name
.  .  .  .  .  request-protection
.  .  .  .  .  response-protection
```

```
.  .  call-property
.  .  .  name
.  .  .  value
.  .  wsdl-override
.  .  service-impl-class
.  .  service-qname
.  .  .  namespaceURI
.  .  .  localpart
.  message-destination-ref
.  .  message-destination-ref-name
.  .  jndi-name
.  cache
.  .  cache-helper
.  .  .  property (with attributes)
.  .  .  .  description
.  .  default-helper
.  .  .  property (with attributes)
.  .  .  .  description
.  .  property (with attributes)
.  .  .  description
.  .  cache-mapping
.  .  .  servlet-name
.  .  .  url-pattern
.  .  .  cache-helper-ref
.  .  .  dispatcher
.  .  .  timeout
.  .  .  refresh-field
.  .  .  http-method
.  .  .  key-field
.  .  .  constraint-field
.  .  .  .  constraint-field-value
.  class-loader
.  .  property (with attributes)
.  .  .  description
.  jsp-config
.  locale-charset-info
.  .  locale-charset-map
.  .  parameter-encoding
.  parameter-encoding
.  property (with attributes)
.  .  description
.  valve
.  message-destination
.  .  message-destination-name
.  .  jndi-name
.  webservice-description
.  .  webservice-description-name
.  .  wsdl-publish-location
.  keep-state
.  version-identifier
```

Here is a sample `glassfish-web.xml` file:

```
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
   <session-config>
      <session-manager/>
   </session-config>
   <resource-ref>
      <res-ref-name>mail/Session</res-ref-name>
      <jndi-name>mail/Session</jndi-name>
   </resource-ref>
   <jsp-config/>
</glassfish-web-app>
```

The glassfish-ejb-jar.xml File

The `glassfish-ejb-jar.xml` file configures an enterprise bean (EJB JAR file). The element hierarchy is as follows:

```
glassfish-ejb-jar
.  security-role-mapping
.  .  role-name
.  .  principal-name
.  .  group-name
.  enterprise-beans
.  .  name
.  .  unique-id
.  .  ejb
.  .  .  ejb-name
.  .  .  jndi-name
.  .  .  ejb-ref
.  .  .  .  ejb-ref-name
.  .  .  .  jndi-name
.  .  .  resource-ref
.  .  .  .  res-ref-name
.  .  .  .  jndi-name
.  .  .  .  default-resource-principal
.  .  .  .  .  name
.  .  .  .  .  password
.  .  .  resource-env-ref
.  .  .  .  resource-env-ref-name
.  .  .  .  jndi-name
.  .  .  service-ref
.  .  .  .  service-ref-name
```

```
. . . . port-info
. . . . . service-endpoint-interface
. . . . . wsdl-port
. . . . . . namespaceURI
. . . . . . localpart
. . . . . stub-property
. . . . . . name
. . . . . . value
. . . . . call-property
. . . . . . name
. . . . . . value
. . . . . message-security-binding
. . . . . . message-security
. . . . . . . message
. . . . . . . . java-method
. . . . . . . . . method-name
. . . . . . . . . method-params
. . . . . . . . . . method-param
. . . . . . . . operation-name
. . . . . . . request-protection
. . . . . . . response-protection
. . . . call-property
. . . . . name
. . . . . value
. . . . wsdl-override
. . . . service-impl-class
. . . . service-qname
. . . . . namespaceURI
. . . . . localpart
. . . message-destination-ref
. . . . message-destination-ref-name
. . . . jndi-name
. . . pass-by-reference
. . . cmp
. . . . mapping-properties
. . . . is-one-one-cmp
. . . . one-one-finders
. . . . . finder
. . . . . . method-name
. . . . . . query-params
. . . . . . query-filter
. . . . . . query-variables
. . . . . . query-ordering
. . . . prefetch-disabled
. . . . . query-method
. . . . . . method-name
. . . . . . method-params
. . . . . . . method-param
. . . principal
. . . . name
. . . mdb-connection-factory
```

```
. . . . jndi-name
. . . . default-resource-principal
. . . . . name
. . . . . password
. . . jms-durable-subscription-name
. . . jms-max-messages-load
. . . ior-security-config
. . . . transport-config
. . . . . integrity
. . . . . confidentiality
. . . . . establish-trust-in-target
. . . . . establish-trust-in-client
. . . . as-context
. . . . . auth-method
. . . . . realm
. . . . . required
. . . . sas-context
. . . . . caller-propagation
. . . is-read-only-bean
. . . refresh-period-in-seconds
. . . commit-option
. . . cmt-timeout-in-seconds
. . . use-thread-pool-id
. . . gen-classes
. . . . remote-impl
. . . . local-impl
. . . . remote-home-impl
. . . . local-home-impl
. . . bean-pool
. . . . steady-pool-size
. . . . resize-quantity
. . . . max-pool-size
. . . . pool-idle-timeout-in-seconds
. . . . max-wait-time-in-millis
. . . bean-cache
. . . . max-cache-size
. . . . resize-quantity
. . . . is-cache-overflow-allowed
. . . . cache-idle-timeout-in-seconds
. . . . removal-timeout-in-seconds
. . . . victim-selection-policy
. . . mdb-resource-adapter
. . . . resource-adapter-mid
. . . . activation-config
. . . . . description
. . . . . activation-config-property
. . . . . . activation-config-property-name
. . . . . . activation-config-property-value
. . . webservice-endpoint
. . . . port-component-name
. . . . endpoint-address-uri
```

```
. . . . login-config
. . . . . auth-method
. . . . . realm
. . . . message-security-binding
. . . . . message-security
. . . . . . message
. . . . . . . java-method
. . . . . . . . method-name
. . . . . . . . method-params
. . . . . . . . . method-param
. . . . . . . operation-name
. . . . . . request-protection
. . . . . . response-protection
. . . . transport-guarantee
. . . . service-qname
. . . . tie-class
. . . . servlet-impl-class
. . . . debugging-enabled
. . . . property (with subelements)
. . . . . name
. . . . . value
. . . flush-at-end-of-method
. . . . method
. . . . . description
. . . . . ejb-name
. . . . . method-name
. . . . . method-intf
. . . . . method-params
. . . . . . method-param
. . . checkpointed-methods
. . . checkpoint-at-end-of-method
. . . . method
. . . . . description
. . . . . ejb-name
. . . . . method-name
. . . . . method-intf
. . . . . method-params
. . . . . . method-param
. . . per-request-load-balancing
. . pm-descriptors
. . cmp-resource
. . . jndi-name
. . . default-resource-principal
. . . . name
. . . . password
. . . property (with subelements)
. . . . name
. . . . value
. . . create-tables-at-deploy
. . . drop-tables-at-undeploy
. . . database-vendor-name
```

```
.  .  .  schema-generator-properties
.  .  .  .  property (with subelements)
.  .  .  .  .  name
.  .  .  .  .  value
.  .  message-destination
.  .  .  message-destination-name
.  .  .  jndi-name
.  .  webservice-description
.  .  .  webservice-description-name
.  .  .  wsdl-publish-location
.  .  property (with subelements)
.  .  .  name
.  .  .  value
.  compatibility
.  disable-nonportable-jndi-names
.  keep-state
.  version-identifier
```

If any configuration information for an enterprise bean is not specified in the `glassfish-ejb-jar.xml` file, it defaults to a corresponding setting in the EJB container if an equivalency exists.

Here is a sample `glassfish-ejb-jar.xml` file:

```
<!DOCTYPE glassfish-ejb-jar PUBLIC "-//GlassFish.org//
DTD GlassFish Application Server 3.1 EJB 3.1//EN"
"http://glassfish.org/dtds/glassfish-ejb-jar_3_1-1.dtd">
<glassfish-ejb-jar>
<display-name>First Module</display-name>
<enterprise-beans>
    <ejb>
        <ejb-name>CustomerEJB</ejb-name>
        <jndi-name>customer</jndi-name>
        <bean-pool>
            <steady-pool-size>10</steady-pool-size>
            <resize-quantity>10</resize-quantity>
            <max-pool-size>100</max-pool-size>
            <pool-idle-timeout-in-seconds>600</pool-idle-timeout-in-seconds>
        </bean-pool>
        <bean-cache>
            <max-cache-size>100</max-cache-size>
            <resize-quantity>10</resize-quantity>
            <removal-timeout-in-seconds>3600</removal-timeout-in-seconds>
            <victim-selection-policy>LRU</victim-selection-policy>
        </bean-cache>
    </ejb>
    <cmp-resource>
        <jndi-name>jdbc/__default</jndi-name>
        <create-tables-at-deploy>true</create-tables-at-deploy>
```

```
        <drop-tables-at-undeploy>true</drop-tables-at-undeploy>
    </cmp-resource>
</enterprise-beans>
<keep-state>true</keep-state>
</glassfish-ejb-jar>
```

The sun-cmp-mappings.xml File

The `sun-cmp-mappings.xml` file configures container-managed persistence for an EJB 2.0 or 2.1 entity bean. The element hierarchy is as follows:

```
sun-cmp-mappings
.  sun-cmp-mapping
.  .  schema
.  .  entity-mapping
.  .  .  ejb-name
.  .  .  table-name
.  .  .  cmp-field-mapping
.  .  .  .  field-name
.  .  .  .  column-name
.  .  .  .  read-only
.  .  .  .  fetched-with
.  .  .  .  .  default
.  .  .  .  .  level
.  .  .  .  .  named-group
.  .  .  .  .  none
.  .  .  cmr-field-mapping
.  .  .  .  cmr-field-name
.  .  .  .  column-pair
.  .  .  .  .  column-name
.  .  .  .  fetched-with
.  .  .  .  .  default
.  .  .  .  .  level
.  .  .  .  .  named-group
.  .  .  .  .  none
.  .  .  secondary-table
.  .  .  .  table-name
.  .  .  .  column-pair
.  .  .  .  .  column-name
.  .  .  consistency
.  .  .  .  none
.  .  .  .  check-modified-at-commit
.  .  .  .  lock-when-loaded
.  .  .  .  check-all-at-commit
.  .  .  .  lock-when-modified
.  .  .  .  check-version-of-accessed-instances
```

```
. . . . . column-name
```

Here is a sample database schema definition:

```
create table TEAMEJB (
   TEAMID varchar2(256) not null,
   NAME varchar2(120) null,
   CITY char(30) not null,
   LEAGUEEJB_LEAGUEID varchar2(256) null,
   constraint PK_TEAMEJB primary key (TEAMID)
)
create table PLAYEREJB (
   POSITION varchar2(15) null,
   PLAYERID varchar2(256) not null,
   NAME char(64) null,
   SALARY number(10, 2) not null,
   constraint PK_PLAYEREJB primary key (PLAYERID)
)
create table LEAGUEEJB (
   LEAGUEID varchar2(256) not null,
   NAME varchar2(256) null,
   SPORT varchar2(256) null,
   constraint PK_LEAGUEEJB primary key (LEAGUEID)
)
create table PLAYEREJBTEAMEJB (
   PLAYEREJB_PLAYERID varchar2(256) null,
   TEAMEJB_TEAMID varchar2(256) null
)
alter table TEAMEJB
   add constraint FK_LEAGUE foreign key (LEAGUEEJB_LEAGUEID)
   references LEAGUEEJB (LEAGUEID)

alter table PLAYEREJBTEAMEJB
   add constraint FK_TEAMS foreign key (PLAYEREJB_PLAYERID)
   references PLAYEREJB (PLAYERID)

alter table PLAYEREJBTEAMEJB
   add constraint FK_PLAYERS foreign key (TEAMEJB_TEAMID)
   references TEAMEJB (TEAMID)
```

Here is a corresponding sample `sun-cmp-mappings.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sun-cmp-mappings>
<sun-cmp-mapping>
    <schema>Roster</schema>
    <entity-mapping>
        <ejb-name>TeamEJB</ejb-name>
        <table-name>TEAMEJB</table-name>
```

```
        <cmp-field-mapping>
            <field-name>teamId</field-name>
            <column-name>TEAMEJB.TEAMID</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>name</field-name>
            <column-name>TEAMEJB.NAME</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>city</field-name>
            <column-name>TEAMEJB.CITY</column-name>
        </cmp-field-mapping>
        <cmr-field-mapping>
            <cmr-field-name>league</cmr-field-name>
            <column-pair>
                <column-name>TEAMEJB.LEAGUEEJB_LEAGUEID</column-name>
                <column-name>LEAGUEEJB.LEAGUEID</column-name>
            </column-pair>
            <fetched-with>
                <none/>
            </fetched-with>
        </cmr-field-mapping>
        <cmr-field-mapping>
            <cmr-field-name>players</cmr-field-name>
            <column-pair>
                <column-name>TEAMEJB.TEAMID</column-name>
                <column-name>PLAYEREJBTEAMEJB.TEAMEJB_TEAMID</column-name>
            </column-pair>
            <column-pair>
                <column-name>PLAYEREJBTEAMEJB.PLAYEREJB_PLAYERID</column-name>
                <column-name>PLAYEREJB.PLAYERID</column-name>
            </column-pair>
            <fetched-with>
                <none/>
            </fetched-with>
        </cmr-field-mapping>
    </entity-mapping>
    <entity-mapping>
        <ejb-name>PlayerEJB</ejb-name>
        <table-name>PLAYEREJB</table-name>
        <cmp-field-mapping>
            <field-name>position</field-name>
            <column-name>PLAYEREJB.POSITION</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>playerId</field-name>
            <column-name>PLAYEREJB.PLAYERID</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>name</field-name>
            <column-name>PLAYEREJB.NAME</column-name>
```

```
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>salary</field-name>
            <column-name>PLAYEREJB.SALARY</column-name>
        </cmp-field-mapping>
        <cmr-field-mapping>
            <cmr-field-name>teams</cmr-field-name>
            <column-pair>
                <column-name>PLAYEREJB.PLAYERID</column-name>
                <column-name>PLAYEREJBTEAMEJB.PLAYEREJB_PLAYERID</column-name>
            </column-pair>
            <column-pair>
                <column-name>PLAYEREJBTEAMEJB.TEAMEJB_TEAMID</column-name>
                <column-name>TEAMEJB.TEAMID</column-name>
            </column-pair>
            <fetched-with>
                <none/>
            </fetched-with>
        </cmr-field-mapping>
    </entity-mapping>
    <entity-mapping>
        <ejb-name>LeagueEJB</ejb-name>
        <table-name>LEAGUEEJB</table-name>
        <cmp-field-mapping>
            <field-name>leagueId</field-name>
            <column-name>LEAGUEEJB.LEAGUEID</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>name</field-name>
            <column-name>LEAGUEEJB.NAME</column-name>
        </cmp-field-mapping>
        <cmp-field-mapping>
            <field-name>sport</field-name>
            <column-name>LEAGUEEJB.SPORT</column-name>
        </cmp-field-mapping>
        <cmr-field-mapping>
            <cmr-field-name>teams</cmr-field-name>
            <column-pair>
                <column-name>LEAGUEEJB.LEAGUEID</column-name>
                <column-name>TEAMEJB.LEAGUEEJB_LEAGUEID</column-name>
            </column-pair>
            <fetched-with>
                <none/>
            </fetched-with>
        </cmr-field-mapping>
    </entity-mapping>
</sun-cmp-mapping>
</sun-cmp-mappings>
```

The glassfish-application-client.xml file

The `glassfish-application-client.xml` file configures an Application Client Container (ACC) client (JAR file). The element hierarchy is as follows:

```
glassfish-application-client
.  ejb-ref
.  .  ejb-ref-name
.  .  jndi-name
.  resource-ref
.  .  res-ref-name
.  .  jndi-name
.  .  default-resource-principal
.  .  .  name
.  .  .  password
.  resource-env-ref
.  .  resource-env-ref-name
.  .  jndi-name
.  service-ref
.  .  service-ref-name
.  .  port-info
.  .  .  service-endpoint-interface
.  .  .  wsdl-port
.  .  .  .  namespaceURI
.  .  .  .  localpart
.  .  .  stub-property
.  .  .  .  name
.  .  .  .  value
.  .  .  call-property
.  .  .  .  name
.  .  .  .  value
.  .  .  message-security-binding
.  .  .  .  message-security
.  .  .  .  .  message
.  .  .  .  .  .  java-method
.  .  .  .  .  .  .  method-name
.  .  .  .  .  .  .  method-params
.  .  .  .  .  .  .  .  method-param
.  .  .  .  .  .  operation-name
.  .  .  .  .  request-protection
.  .  .  .  .  response-protection
.  .  call-property
.  .  .  name
.  .  .  value
.  .  wsdl-override
.  .  service-impl-class
.  .  service-qname
.  .  .  namespaceURI
.  .  .  localpart
.  message-destination-ref
```

```
.  .  message-destination-ref-name
.  .  jndi-name
.  message-destination
.  .  message-destination-name
.  .  jndi-name
.  java-web-start-access
.  .  context-root
.  .  eligible
.  .  vendor
.  .  jnlp-doc
.  version-identifier
```

Here is a sample `glassfish-application-client.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-application-client PUBLIC ""-//GlassFish.org//DTD
GlassFish Application Server 3.1 Application Client 6.0//EN""
"http://glassfish.org/dtds/glassfish-application-client_6_0-1.dtd">
<glassfish-application-client>
  <message-destination-ref>
    <message-destination-ref-name>ClientQueue</message-destination-ref-name>
    <jndi-name>jms/security_mdb_OutQueue</jndi-name>
  </message-destination-ref>
</glassfish-application-client>
```

The sun-acc.xml File

The `sun-acc.xml` file configures the Application Client Container. This is more of a configuration file than a deployment descriptor. Eclipse GlassFish provides a default file in the domain-dir`/config` directory. Specifying a different file is optional. The element hierarchy is as follows:

```
client-container
.  target-server
.  .  description
.  .  security
.  .  .  ssl
.  .  .  cert-db
.  auth-realm
.  .  property (with attributes)
.  client-credential
.  .  property (with attributes)
.  log-service
.  .  property (with attributes)
.  message-security-config
.  .  provider-config
.  .  .  request-policy
```

```
.  .  .  response-policy
.  .  .  property (with attributes)
.  property (with attributes)
```

The glassfish-resources.xml File

The `glassfish-resources.xml` file configures application-scoped resources. The element hierarchy is as follows:

```
resources
.  custom-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  external-jndi-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  jdbc-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  mail-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  admin-object-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  connector-resource
.  .  description
.  .  property (with attributes)
.  .  .  description
.  resource-adapter-config
.  .  property (with attributes)
.  .  .  description
.  jdbc-connection-pool
.  .  description
.  .  property (with attributes)
.  .  .  description
.  connector-connection-pool
.  .  description
.  .  security-map
.  .  .  principal
.  .  .  user-group
.  .  .  backend-principal
```

```
.  .  property (with attributes)
.  .  .  description
.  work-security-map
.  .  description
.  .  principal-map
.  .  group-map
```

WebLogic Server Deployment Descriptor Support in Eclipse GlassFish

Eclipse GlassFish offers limited support for the `weblogic-application.xml`, `weblogic.xml`, and `weblogic-webservices.xml` deployment descriptor files.

The only element in `weblogic-application.xml` that Eclipse GlassFish supports is `security`. The equivalent element in the `glassfish-application.xml` file is `security-role-mapping`.

The elements of `weblogic.xml` that Eclipse GlassFish supports are explained in the following table.

Table B-2 `weblogic.xml` Support in Eclipse GlassFish

`weblogic.xml` **Element Name**	**Eclipse GlassFish Support**
`role-name` under `security-role-assignment`	`role-name` under `security-role-mapping` `glassfish-web.xml` equivalent
`principal-name` under `security-role-assignment`	`principal-name` under `security-role-mapping` `glassfish-web.xml` equivalent
`resource-description`	`resource-ref` `glassfish-web.xml` equivalent, but `resource-link` not supported
`resource-env-description`	`resource-env-ref` `glassfish-web.xml` equivalent, but `resource-link` not supported
`ejb-reference-description`	`ejb-ref` `glassfish-web.xml` equivalent
`service-reference-description`	`service-ref` `glassfish-web.xml` equivalent
`timeout-secs` under `session-descriptor`	`timeoutSeconds` property of `session-properties` `glassfish-web.xml` equivalent
`invalidation-interval-secs` under `session-descriptor`	`reapIntervalSeconds` property of `manager-properties` `glassfish-web.xml` equivalent
`max-in-memory-sessions` under `session-descriptor`	`maxSessions` property of `manager-properties` `glassfish-web.xml` equivalent
`persistent-store-dir` under `session-descriptor`	`directory` property of `store-properties` `glassfish-web.xml` equivalent
`prefer-web-inf-classes` under `container-descriptor`	`delegate` attribute of `class-loader` `glassfish-web.xml` equivalent
`context-root`	`context-root` `glassfish-web.xml` equivalent

weblogic.xml Element Name	Eclipse GlassFish Support
cookies-enabled under session-descriptor	Servlet 3.0
cookie-name under session-descriptor	Servlet 3.0
cookie-path under session-descriptor	Servlet 3.0
cookie-domain under session-descriptor	Servlet 3.0
cookie-comment under session-descriptor	Servlet 3.0
cookie-secure under session-descriptor	Servlet 3.0
cookie-max-age-secs under session-descriptor	Servlet 3.0
cookie-http-only under session-descriptor	Servlet 3.0
url-rewriting-enabled under session-descriptor	Servlet 3.0
persistent-store-cookie-name under session-descriptor	Cookie-based persistence is supported
keepgenerated under jsp-descriptor	keepgenerated init parameter of JspServlet
working-dir under jsp-descriptor	scratchdir init parameter of JspServlet
compress-html-template under jsp-descriptor	trimSpaces init parameter of JspServlet
index-directory-enabled under container-descriptor	listings init parameter of DefaultServlet
index-directory-sort-by under container-descriptor	sortedBy init parameter of DefaultServlet
save-sessions-enabled under container-descriptor	Same as asadmin redeploy --keepstate=true or keep-state in glassfish-web.xml
run-as-principal-name under servlet-descriptor	principal-name under servlet glassfish-web.xml equivalent

The elements of weblogic-webservices.xml that Eclipse GlassFish supports are explained in the following table.

Table B-3 weblogic-webservices.xml Support in Eclipse GlassFish

weblogic-webservices.xml Element Name	Eclipse GlassFish Support
webservice-type	Possible values are JAXRPC or JAXWS. Eclipse GlassFish does not support JAX-RPC web services with JSR 181 annotations. The use of this element is limited, because the container can find out if the type is JAX-WS or JAX-RPC based on presence of JSR 181 annotations.
wsdl-publish-file	Same as wsdl-publish-location in glassfish-web.xml

`weblogic-webservices.xml` **Element Name**	**Eclipse GlassFish Support**
`service-endpoint-address`	Similar to `endpoint-address-uri` in `glassfish-web.xml`, except that `webservice-contextpath` and `webservice-serviceuri` are specified separately
`j2ee:login-config`	Same as `login-config` in `glassfish-web.xml`
`j2ee:transport-guarantee`	Same as `transport-guarantee` in `glassfish-web.xml`
`exposed` under `wsdl`	Accepts `true` or `false`, defaults to `true`. Controls the publishing of WSDL to clients.
`stream-attachments`	Accepts `true` or `false`, defaults to `true`. Only for JAX-WS web services. Configures the JAX-WS runtime to send attachments in streaming fashion.
`validate-request`	Accepts `true` or `false`, defaults to `false`. Only for JAX-WS web services. Configures the JAX-WS runtime to validate that request messages are as the WSDL definitions specify.
`http-response-buffersize`	Property of `ReliabilityMessagingFeature` configuration, similar to `ReliableMessagingFeature.setDestinationBufferQuota()`
`reliability-config`	Partially supported. Subelements map to Metro's `ReliabilityMessagingFeature`.
`inactivity-timeout` under `reliability-config`	Maps to `ReliableMessagingFeature.getSequenceInactivityTimeout()`
`base-retransmission-interval` under `reliability-config`	Maps to `ReliableMessagingFeature.``getMessageRetransmissionInterval()`
`retransmission-exponential-backoff` under `reliability-config`	Maps to `ReliableMessagingFeature.``getRetransmissionBackoffAlgorithm()`. Returns enum values, one of them is `exponential`.
`acknowledgement-interval` under `reliability-config`	Maps to `ReliableMessagingFeature.``getAcknowledgementTransmissionInterval()`
`sequence-expiration` under `reliability-config`	Maps to `ReliableMessagingFeature.``getSequenceInactivityTimeout()`. In WebLogic Server this value applies regardless of activity. In Metro it applies only to inactive sequences.
`buffer-retry-count` under `reliability-config`	Maps to `ReliableMessagingFeature.``getMaxMessageRetransmissionCount()`
`buffer-retry-delay` under `reliability-config`	Maps to `ReliableMessagingFeature.``getMessageRetransmissionInterval()`

C Elements of the Eclipse GlassFish Deployment Descriptors

This appendix describes the elements of the Eclipse GlassFishdeployment descriptors.

activation-config

Specifies an activation configuration, which includes the runtime configuration properties of the message-driven bean in its operational environment. For example, this can include information about the name of a physical JMS destination. Matches and overrides the `activation-config` element in the `ejb-jar.xml` file.

Superelements

`mdb-resource-adapter` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `activation-config` element.

Table C-1 `activation-config` subelements

Element	Required	Description
`description`	zero or one	Specifies a text description of the activation configuration.
`activation-config-property`	one or more	Specifies an activation configuration property.

activation-config-property

Specifies the name and value of an activation configuration property.

Superelements

`activation-config` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `activation-config-property` element.

Table C-2 `activation-config-property` subelements

Element	Required	Description
`activation-config-property-name`	only one	Specifies the name of an activation configuration property.

Element	Required	Description
activation-config-property-value	only one	Specifies the value of an activation configuration property.

activation-config-property-name

Specifies the name of an activation configuration property.

Superelements

activation-config-property (glassfish-ejb-jar.xml)

Subelements

none - contains data

activation-config-property-value

Specifies the value of an activation configuration property.

Superelements

activation-config-property (glassfish-ejb-jar.xml)

Subelements

none - contains data

admin-object-resource

Defines an administered object for an inbound resource adapter.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the admin-object-resource element.

Table C-3 admin-object-resource Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.

Element	Required	Description
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the admin-object-resource element.

Table C-4 admin-object-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
res-type	none	Specifies the fully qualified type of the resource.
res-adapter	none	Specifies the name of the inbound resource adapter.
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

Properties

Properties of the admin-object-resource element are the names of setter methods of the class referenced by the adminobject-class of the ra.xml file. Some of the property names can be specified in the adminobjectType element.

as-context

Specifies the authentication mechanism used to authenticate the client.

Superelements

ior-security-config (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the `as-context` element.

Table C-5 `as-context` Subelements

Element	Required	Description
`auth-method`	only one	Specifies the authentication method. The only supported value is `USERNAME_PASSWORD`.
`realm`	only one	Specifies the realm in which the user is authenticated.
`required`	only one	Specifies whether the authentication method specified in the `auth-method` element must be used for client authentication.

`archive-name`

Specifies the name of the archive file. The value of the `archive-name` element is used to derive the default application name when `display-name` is not present in the `application.xml` file. The default application name is the `archive-name` value minus the file extension. For example, if `archive-name` is `foo.ear`, the default application name is `foo`.

Superelements

`glassfish-application` (`glassfish-application.xml`)

Subelements

none - contains data

`auth-method`

Specifies the authentication method.

If the parent element is `as-context`, the only supported value is `USERNAME_PASSWORD`.

If the parent element is `login-config`, specifies the authentication mechanism for the web service endpoint. As a prerequisite to gaining access to any web resources protected by an authorization constraint, a user must be authenticated using the configured mechanism.

Superelements

`login-config` (`glassfish-web.xml`), `as-context` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

auth-realm

JAAS is available on the ACC. Defines the optional configuration for a JAAS authentication realm. Authentication realms require provider-specific properties, which vary depending on what a particular implementation needs. For more information about how to define realms, see "Realm Configuration" in Eclipse GlassFish Application Development Guide.

Superelements

`client-container` (`sun-acc.xml`)

Subelements

The following table describes subelements for the `auth-realm` element.

Table C-6 `auth-realm` subelement

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the `auth-realm` element.

Table C-7 `auth-realm` attributes

Attribute	Default	Description
`name`	none	Defines the name of this realm.
`classname`	none	Defines the Java class which implements this realm.

Example

Here is an example of the default file realm:

```
<auth-realm name="file"
   classname="com.sun.enterprise.security.auth.realm.file.FileRealm">
   <property name="file" value="domain-dir/config/keyfile"/>
   <property name="jaas-context" value="fileRealm"/>
</auth-realm>
```

Which properties an `auth-realm` element uses depends on the value of the `auth-realm` element's `name` attribute. The file realm uses `file` and `jaas-context` properties. Other realms use different properties. See "Realm Configuration" in Eclipse GlassFish Application Development Guide.

backend-principal

Specifies the user name and password required by the Enterprise Information System (EIS).

Superelements

`security-map` (`glassfish-resources.xml`)

Subelements

none

Attributes

The following table describes attributes for the `backend-principal` element.

Table C-8 `backend-principal` Attributes

Attribute	Default	Description
`user-name`	none	Specifies the user name required by the EIS.
`password`	none	(optional) Specifies the password required by the EIS, if any.

bean-cache

Specifies the entity bean cache properties. Used for entity beans and stateful session beans.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `bean-cache` element.

Table C-9 `bean-cache` Subelements

Element	Required	Description
`max-cache-size`	zero or one	Specifies the maximum number of beans allowable in cache.
`is-cache-overflow-allowed`	zero or one	Deprecated.
`cache-idle-timeout-in-seconds`	zero or one	Specifies the maximum time that a stateful session bean or entity bean is allowed to be idle in cache before being passivated. Default value is 10 minutes (600 seconds).

Element	Required	Description
removal-timeout-in-seconds	zero or one	Specifies the amount of time a bean remains before being removed. If removal-timeout-in-seconds is less than idle-timeout, the bean is removed without being passivated.
resize-quantity	zero or one	Specifies the number of beans to be created if the pool is empty (subject to the max-pool-size limit). Values are from 0 to MAX_INTEGER.
victim-selection-policy	zero or one	Specifies the algorithm that must be used by the container to pick victims. Applies only to stateful session beans.

Example

```
<bean-cache>
   <max-cache-size>100</max-cache-size>
   <cache-resize-quantity>10</cache-resize-quantity>
   <removal-timeout-in-seconds>3600</removal-timeout-in-seconds>
   <victim-selection-policy>LRU</victim-selection-policy>
      <cache-idle-timeout-in-seconds>600</cache-idle-timeout-in-seconds>
   <removal-timeout-in-seconds>5400</removal-timeout-in-seconds>
</bean-cache>
```

bean-pool

Specifies the pool properties of stateless session beans, entity beans, and message-driven bean.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the bean-pool element.

Table C-10 bean-pool Subelements

Element	Required	Description
steady-pool-size	zero or one	Specifies the initial and minimum number of beans maintained in the pool. Default is 32.
resize-quantity	zero or one	Specifies the number of beans to be created if the pool is empty (subject to the max-pool-size limit). Values are from 0 to MAX_INTEGER.

Element	Required	Description
max-pool-size	zero or one	Specifies the maximum number of beans in the pool. Values are from 0 to MAX_INTEGER. Default is to the EJB container value or 60.
max-wait-time-in-millis	zero or one	Deprecated.
pool-idle-timeout-in-seconds	zero or one	Specifies the maximum time that a bean is allowed to be idle in the pool. After this time, the bean is removed. This is a hint to the server. Default time is 600 seconds (10 minutes).

Example

```
<bean-pool>
   <steady-pool-size>10</steady-pool-size>
   <resize-quantity>10</resize-quantity>
   <max-pool-size>100</max-pool-size>
   <pool-idle-timeout-in-seconds>600</pool-idle-timeout-in-seconds>
</bean-pool>
```

cache

Configures caching for web application components.

Superelements

glassfish-web-app (glassfish-web.xml)

Subelements

The following table describes subelements for the cache element.

Table C-11 cache Subelements

Element	Required	Description
cache-helper	zero or more	Specifies a custom class that implements the CacheHelper interface.
default-helper	zero or one	Allows you to change the properties of the default, built-in cache-helper class.
property (with attributes)	zero or more	Specifies a cache property, which has a name and a value.
cache-mapping	zero or more	Maps a URL pattern or a servlet name to its cacheability constraints.

Attributes

The following table describes attributes for the `cache` element.

Table C-12 `cache` Attributes

Attribute	Default	Description
`max-entries`	`4096`	(optional) Specifies the maximum number of entries the cache can contain. Must be a positive integer.
`timeout-in-seconds`	`30`	(optional) Specifies the maximum amount of time in seconds that an entry can remain in the cache after it is created or refreshed. Can be overridden by a `timeout` element.
`enabled`	`true`	(optional) Determines whether servlet and JSP caching is enabled.

Properties

The following table describes properties for the `cache` element.

Table C-13 `cache` Properties

Property	Default	Description
`cacheClassName`	`com.sun.appserv.web.cache.LruCache`	Specifies the fully qualified name of the class that implements the cache functionality. See Cache Class Names for possible values.
`MultiLRUSegmentSize`	`4096`	Specifies the number of entries in a segment of the cache table that should have its own LRU (least recently used) list. Applicable only if `cacheClassName` is set to `com.sun.appserv.web.cache.MultiLruCache`.
`MaxSize`	unlimited; `Long.MAX_VALUE`	Specifies an upper bound on the cache memory size in bytes (KB or MB units). Example values are `32 KB` or `2 MB`. Applicable only if `cacheClassName` is set to `com.sun.appserv.web.cache.BoundedMultiLruCache`.

Cache Class Names

The following table lists possible values of the `cacheClassName` property.

Table C-14 `cacheClassName` Values

Value	Description
com.sun.appserv.web.cache.LruCache	A bounded cache with an LRU (least recently used) cache replacement policy.
com.sun.appserv.web.cache.BaseCache	An unbounded cache suitable if the maximum number of entries is known.
com.sun.appserv.web.cache.MultiLruCache	A cache suitable for a large number of entries (>4096). Uses the MultiLRUSegmentSize property.
com.sun.appserv.web.cache.BoundedMultiLruCache	A cache suitable for limiting the cache size by memory rather than number of entries. Uses the MaxSize property.

cache-helper

Specifies a class that implements the com.sun.appserv.web.cache.CacheHelper interface.

Superelements

cache (glassfish-web.xml)

Subelements

The following table describes subelements for the cache-helper element.

Table C-15 cache-helper Subelements

Element	Required	Description
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the cache-helper element.

Table C-16 cache-helper Attributes

Attribute	Default	Description
name	default	Specifies a unique name for the helper class, which is referenced in the cache-mapping element.
class-name	none	Specifies the fully qualified class name of the cache helper, which must implement the com.sun.appserv.web.CacheHelper interface.

cache-helper-ref

Specifies the `name` of the `cache-helper` used by the parent `cache-mapping` element.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none - contains data

cache-idle-timeout-in-seconds

Specifies the maximum time that a bean can remain idle in the cache. After this amount of time, the container can passivate this bean. A value of `0` specifies that beans never become candidates for passivation. Default is 600.

Applies to stateful session beans and entity beans.

Superelements

`bean-cache` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

cache-mapping

Maps a URL pattern or a servlet name to its cacheability constraints.

Superelements

`cache` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `cache-mapping` element.

Table C-17 `cache-mapping` Subelements

Element	Required	Description
`servlet-name`	requires one `servlet-name` or `url-pattern`	Contains the name of a servlet.
`url-pattern`	requires one `servlet-name` or `url-pattern`	Contains a servlet URL pattern for which caching is enabled.

Element	Required	Description
cache-helper-ref	required if dispatcher, timeout, refresh-field, http-method, key-field, and constraint-field are not used	Contains the name of the cache-helper used by the parent cache-mapping element.
dispatcher	zero or one if cache-helper-ref is not used	Contains a comma-separated list of RequestDispatcher methods for which caching is enabled.
timeout	zero or one if cache-helper-ref is not used	Contains the cache-mapping specific maximum amount of time in seconds that an entry can remain in the cache after it is created or refreshed.
refresh-field	zero or one if cache-helper-ref is not used	Specifies a field that gives the application component a programmatic way to refresh a cached entry.
http-method	zero or more if cache-helper-ref is not used	Contains an HTTP method that is eligible for caching.
key-field	zero or more if cache-helper-ref is not used	Specifies a component of the key used to look up and extract cache entries.
constraint-field	zero or more if cache-helper-ref is not used	Specifies a cacheability constraint for the given url-pattern or servlet-name.

call-property

Specifies JAX-RPC property values that can be set on a javax.xml.rpc.Call object before it is returned to the web service client. The property names can be any properties supported by the JAX-RPC Call implementation.

Superelements

port-info, service-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

The following table describes subelements for the call-property element.

Table C-18 call-property subelements

Element	Required	Description
name	only one	Specifies the name of the entity.
value	only one	Specifies the value of the entity.

caller-propagation

Specifies whether the target accepts propagated caller identities. The values are NONE, SUPPORTED, or REQUIRED.

Superelements

sas-context (glassfish-ejb-jar.xml)

Subelements

none - contains data

cert-db

Not implemented. Included for backward compatibility only. Attribute values are ignored.

Superelements

security (sun-acc.xml)

Subelements

none

Attributes

The following table describes attributes for the cert-db element.

Table C-19 cert-db attributes

Attribute	Default	Description
path	none	Specifies the absolute path of the certificate database.
password	none	Specifies the password to access the certificate database.

check-all-at-commit

This element is not implemented. Do not use.

Superelements

consistency (sun-cmp-mappings.xml)

check-modified-at-commit

Checks concurrent modification of fields in modified beans at commit time.

Superelements

`consistency` (`sun-cmp-mappings.xml`)

Subelements

none - element is present or absent

check-version-of-accessed-instances

Checks the version column of the modified beans.

Version consistency allows the bean state to be cached between transactions instead of read from a database. The bean state is verified by primary key and version column values. This occurs during a custom query (for dirty instances only) or commit (for both clean and dirty instances).

The version column must be a numeric type, and must be in the primary table. You must provide appropriate update triggers for this column.

Superelements

`consistency` (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `check-version-of-accessed-instances` element.

Table C-20 `check-version-of-accessed-instances` Subelements

Element	Required	Description
`column-name`	only one	Specifies the name of the version column.

checkpoint-at-end-of-method

Specifies that the stateful session bean state is checkpointed, or persisted, after the specified methods are executed. The `availability-enabled` attribute of the parent `ejb` element must be set to `true`.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `checkpoint-at-end-of-method` element.

Table C-21 `checkpoint-at-end-of-method` Subelements

Element	Required	Description
method	one or more	Specifies a bean method.

checkpointed-methods

Deprecated. Supported for backward compatibility. Use checkpoint-at-end-of-method instead.

Superelements

ejb (glassfish-ejb-jar.xml)

class-loader

Configures the class loader for the web module.

Superelements

glassfish-web-app (glassfish-web.xml)

Subelements

The following table describes subelements for the class-loader element.

Table C-22 class-loader Subelements

Element	Required	Description
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the class-loader element.

Table C-23 class-loader Attributes

Attribute	Default	Description
extra-class-path	null	(optional) Specifies a colon or semicolon separated list of additional classpaths for this web module. Paths can be absolute or relative to the web module's root, for example: `extra-class-path="WEB-INF/lib/extra/extra.jar"`

Attribute	Default	Description
`delegate`	`true`	(optional) If `true`, the web module follows the standard class loader delegation model and delegates to its parent class loader first before looking in the local class loader. You must set this to `true` for a web module that accesses EJB components or that acts as a web service client or endpoint. If `false`, the web module follows the delegation model specified in the Servlet specification and looks in its class loader before looking in the parent class loader. It's safe to set this to `false` only for a web module that does not interact with any other modules. 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 Jakarta EE platform.
`dynamic-reload-interval`		(optional) Not implemented. Included for backward compatibility with previous Oracle Web Server versions.

If the `delegate` attribute is set to `false`, the class loader delegation behavior complies with the Servlet 2.4 specification, section 9.7.2. If set to its default value of `true`, classes and resources residing in container-wide library JAR files are loaded in preference to classes and resources packaged within the WAR file.

Portable programs that use this element should not be packaged with any classes or interfaces that are a part of the Jakarta EE specification. The behavior of a program that includes such classes or interfaces in its WAR file is undefined.

Properties

The following table describes properties for the `class-loader` element.

Table C-24 `class-loader` Properties

Property	Default	Description
`ignoreHiddenJarFiles`	`false`	If `true`, specifies that all JAR and ZIP files in the `WEB-INF/lib` directory that start with a period (`.`) are ignored by the class loader.

`client-container`

Defines the Eclipse GlassFish specific configuration for the application client container. This is the root element; there can only be one `client-container` element in a `sun-acc.xml` file. See The sun-acc.xml File.

Superelements

none

Subelements

The following table describes subelements for the `client-container` element.

Table C-25 `client-container` Subelements

Element	Required	Description
`target-server`	one or more	Specifies the IIOP listener for the target server. Also specifies IIOP endpoints used for load balancing. If the Eclipse GlassFish instance on which the application client is deployed participates in a cluster, Eclipse GlassFish finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed. A listener or endpoint is in the form host`:`port, where the host is an IP address or host name, and the port specifies the port number.
`auth-realm`	zero or one	Specifies the optional configuration for JAAS authentication realm.
`client-credential`	zero or one	Specifies the default client credential that is sent to the server.
`log-service`	zero or one	Specifies the default log file and the severity level of the message.
`message-security-config`	zero or more	Specifies configurations for message security providers.

Element	Required	Description
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the client-container element.

Table C-26 client-container Attributes

Attribute	Default	Description
send-password	true	If true, specifies that client authentication credentials must be sent to the server. Without authentication credentials, all access to protected EJB components results in exceptions.

Properties

The following table describes properties for the client-container element.

Table C-27 client-container Properties

Property	Default	Description
com.sun.appserv.iiop.endpoints	none	Specifies a comma-separated list of one or more IIOP endpoints used for load balancing. An IIOP endpoint is in the form host`:`port, where the host is an IP address or host name, and the port specifies the port number. Deprecated. Use target-server elements instead.

client-credential

Default client credentials that are sent to the server. If this element is present, the credentials are automatically sent to the server, without prompting the user for the user name and password on the client side.

Superelements

client-container (sun-acc.xml)

Subelements

The following table describes subelements for the `client-credential` element.

Table C-28 `client-credential` subelement

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the `client-credential` element.

Table C-29 `client-credential` attributes

Attribute	Default	Description
`user-name`	none	The user name used to authenticate the Application client container.
`password`	none	The password used to authenticate the Application client container.
`realm`	default realm for the domain	(optional) The realm (specified by name) where credentials are to be resolved.

`cmp`

Describes runtime information for a CMP entity bean object for EJB 1.1 and EJB 2.1 beans.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `cmp` element.

Table C-30 `cmp` Subelements

Element	Required	Description
`mapping-properties`	zero or one	This element is not implemented.
`is-one-one-cmp`	zero or one	This element is not implemented.
`one-one-finders`	zero or one	Describes the finders for CMP 1.1 beans.

Element	Required	Description
prefetch-disabled	zero or one	Disables prefetching of entity bean states for the specified query methods.

cmp-field-mapping

The cmp-field-mapping element associates a field with one or more columns to which it maps. The column can be from a bean's primary table or any defined secondary table. If a field is mapped to multiple columns, the column listed first in this element is used as a source for getting the value from the database. The columns are updated in the order they appear. There is one cmp-field-mapping element for each cmp-field element defined in the ejb-jar.xml file.

Superelements

entity-mapping (sun-cmp-mappings.xml)

Subelements

The following table describes subelements for the cmp-field-mapping element.

Table C-31 cmp-field-mapping Subelements

Element	Required	Description
field-name	only one	Specifies the Java identifier of a field. This identifier must match the value of the field-name subelement of the cmp-field that is being mapped.
column-name	one or more	Specifies the name of a column from the primary table, or the qualified table name (TABLE.COLUMN) of a column from a secondary or related table.
read-only	zero or one	Specifies that a field is read-only.
fetched-with	zero or one	Specifies the fetch group for this CMP field's mapping.

cmp-resource

Specifies the database to be used for storing CMP beans. For more information about this element, see "Configuring the CMP Resource" in Eclipse GlassFish Application Development Guide.

Superelements

enterprise-beans (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the cmp-resource element.

Table C-32 `cmp-resource` Subelements

Element	Required	Description
`jndi-name`	only one	Specifies the absolute `jndi-name` of a JDBC resource.
`default-resource-principal`	zero or one	Specifies the default runtime bindings of a resource reference.
`property` (with subelements)	zero or more	Specifies a property name and value. Used to configure `PersistenceManagerFactory` properties.
`create-tables-at-deploy`	zero or one	If `true`, specifies that database tables are created for beans that are automatically mapped by the EJB container.
`drop-tables-at-undeploy`	zero or one	If `true`, specifies that database tables that were automatically created when the bean(s) were last deployed are dropped when the bean(s) are undeployed.
`database-vendor-name`	zero or one	Specifies the name of the database vendor for which tables can be created.
`schema-generator-properties`	zero or one	Specifies field-specific type mappings and allows you to set the `use-unique-table-names` property.

`cmr-field-mapping`

A container-managed relationship field has a name and one or more column pairs that define the relationship. There is one `cmr-field-mapping` element for each `cmr-field` element in the `ejb-jar.xml` file. A relationship can also participate in a fetch group.

Superelements

`entity-mapping` (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `cmr-field-mapping` element.

Table C-33 `cmr-field-mapping` Subelements

Element	Required	Description
`cmr-field-name`	only one	Specifies the Java identifier of a field. Must match the value of the `cmr-field-name` subelement of the `cmr-field` that is being mapped.
`column-pair`	one or more	Specifies the pair of columns that determine the relationship between two database tables.
`fetched-with`	zero or one	Specifies the fetch group for this CMR field's relationship.

cmr-field-name

Specifies the Java identifier of a field. Must match the value of the `cmr-field-name` subelement of the `cmr-field` element in the `ejb-jar.xml` file.

Superelements

`cmr-field-mapping` (`sun-cmp-mappings.xml`)

Subelements

none - contains data

cmt-timeout-in-seconds

Overrides the Transaction Timeout setting of the Transaction Service for an individual bean. The default value, `0`, specifies that the default Transaction Service timeout is used. If positive, this value is used for all methods in the bean that start a new container-managed transaction. This value is not used if the bean joins a client transaction.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

column-name

Specifies the name of a column from the primary table, or the qualified table name (TABLE.COLUMN) of a column from a secondary or related table.

Superelements

`check-version-of-accessed-instances`, `cmp-field-mapping`, `column-pair` (`sun-cmp-mappings.xml`)

Subelements

none - contains data

column-pair

Specifies the pair of columns that determine the relationship between two database tables. Each `column-pair` must contain exactly two `column-name` subelements, which specify the column's names. The first `column-name` element names the table that this bean is mapped to, and the second `column-name` names the column in the related table.

Superelements

`cmr-field-mapping`, `secondary-table` (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `column-pair` element.

Table C-34 `column-pair` Subelements

Element	Required	Description
`column-name`	two	Specifies the name of a column from the primary table, or the qualified table name (TABLE.COLUMN) of a column from a secondary or related table.

`commit-option`

Specifies the commit option used on transaction completion. Valid values for Eclipse GlassFish are `B` or `C`. Default value is `B`. Applies to entity beans.

Commit option A is not supported for this Eclipse GlassFish release.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`compatibility`

Specifies the Eclipse GlassFish release with which to be backward compatible in terms of JAR visibility requirements for applications. The current allowed value is `v2`, which refers to Eclipse GlassFish version 2 or Eclipse GlassFish version 9.1 or 9.1.1. Starting in Jakarta EE 6, the Jakarta EE specification imposes stricter requirements than Jakarta EE 5 did on which JAR files can be visible to various modules within an EAR file. Setting this element to `v2` removes these Jakarta EE 6 and later restrictions.

Superelements

`glassfish-application` (`glassfish-application.xml`), `glassfish-ejb-jar` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

confidentiality

Specifies if the target supports privacy-protected messages. The values are NONE, SUPPORTED, or REQUIRED.

Superelements

transport-config (glassfish-ejb-jar.xml)

Subelements

none - contains data

connector-connection-pool

Defines a connector connection pool.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the connector-connection-pool element.

Table C-35 connector-connection-pool Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
security-map	zero or more	Maps the principal received during servlet or EJB authentication to the credentials accepted by the EIS.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the connector-connection-pool element. Changing the following attributes requires a server restart or the redeployment or disabling and re-enabling of applications that refer to the resource: resource-adapter-name, connection-definition-name, transaction-support, associate-with-thread, lazy-connection-association, and lazy-connection-enlistment.

Table C-36 connector-connection-pool Attributes

Attribute	Default	Description
`name`	none	Specifies the name of the connection pool. A `connector-resource` element's `pool-name` attribute refers to this `name`.
`resource-adapter-name`	none	Specifies the name of the deployed connector module or application. If no name is specified during deployment, the name of the `.rar` file is used. If the resource adapter is embedded in an application, then it is app_name#rar_name.
`connection-definition-name`	none	Specifies a unique name, identifying a resource adapter's `connection-definition` element in the `ra.xml` file. This is usually the `connectionfactory-interface` of the `connection-definition` element.
`steady-pool-size`	`8`	(optional) Specifies the initial and minimum number of connections maintained in the pool.
`max-pool-size`	`32`	(optional) Specifies the maximum number of connections that can be created to satisfy client requests.
`max-wait-time-in-millis`	`60000`	(optional) Specifies the amount of time, in milliseconds, that the caller is willing to wait for a connection. If `0`, the caller is blocked indefinitely until a resource is available or an error occurs.

Attribute	Default	Description
`pool-resize-quantity`	`2`	(optional) Specifies the number of idle connections to be destroyed if the existing number of connections is above the `steady-pool-size` (subject to the `max-pool-size` limit). This is enforced periodically at the `idle-timeout-in-seconds` interval. An idle connection is one that has not been used for a period of `idle-timeout-in-seconds`. When the pool size reaches `steady-pool-size`, connection removal stops.
`idle-timeout-in-seconds`	`300`	(optional) Specifies the maximum time that a connection can remain idle in the pool. After this amount of time, the pool can close this connection.
`fail-all-connections`	`false`	(optional) If `true`, closes all connections in the pool if a single validation check fails.

Attribute	Default	Description
`transaction`-support`	none	(optional) Specifies the transaction support for this connection pool. Overrides the transaction support defined in the resource adapter in a downward compatible way: supports a transaction level lower than or equal to the resource adapter's, but not higher. Allowed values in descending order are: • `XATransaction` - Supports distributed transactions. • `LocalTransaction` - Supports local transactions only. • `NoTransaction` - No transaction support.
`is-connection-validation-required`	`false`	(optional) Specifies whether connections have to be validated before being given to the application. If a resource's validation fails, it is destroyed, and a new resource is created and returned.
`validate-atmost-once-period-in-seconds`	`0`	Specifies the time interval within which a connection is validated at most once. Minimizes the number of validation calls. A value of zero allows unlimited validation calls.

Attribute	Default	Description
`connection-leak-timeout-in-seconds`	`0`	Detects potential connection leaks by the application. A connection that is not returned back to the pool by the application within the specified period is assumed to be potentially leaking, and a stack trace of the caller is logged. A zero value disables leak detection. A nonzero value enables leak tracing.
`connection-leak-reclaim`	`false`	If `true`, the pool will reclaim a connection after `connection-leak-timeout-in-seconds` occurs.
`connection-creation-retry-attempts`	`0`	Specifies the number of attempts to create a new connection.
`connection-creation-retry-interval-in-seconds`	`10`	Specifies the time interval between attempts to create a connection when `connection-creation-retry-attempts` is greater than `0`.
`lazy-connection-enlistment`	`false`	If `true`, a connection is not enlisted in a transaction until it is used. If `false`, any connection object available to a transaction is enlisted in the transaction.
`lazy-connection-association`	`false`	If `true`, a physical connection is not associated with a logical connection until it is used. If `false`, a physical connection is associated with a logical connection even before it is used.

Attribute	Default	Description
associate-with-thread	false	If true, allows connections to be saved as ThreadLocal in the calling thread. Connections get reclaimed only when the calling thread dies or when the calling thread is not in use and the pool has run out of connections. If false, the thread must obtain a connection from the pool each time the thread requires a connection.
		This attribute associates connections with a thread such that when the same thread is in need of connections, it can reuse the connections already associated with that thread. In this case, the overhead of getting connections from the pool is avoided. However, when this value is set to true, you should verify that the value of the max-pool-size attribute is comparable to the max-thread-pool-size attribute of the associated thread pool. If the max-thread-pool-size value is much higher than the max-pool-size value, a lot of time is spent associating connections with a new thread after dissociating them from an older one. Use this attribute in cases where the thread pool should reuse connections to avoid this overhead.

Attribute	Default	Description
match-connections	true	If true, enables connection matching. You can set to false if connections are homogeneous.
max-connection-usage-count	0	Specifies the number of times a connections is reused by the pool, after which it is closed. A zero value disables this feature.
ping	false	(optional) Specifies whether to ping the pool during pool creation or reconfiguration to identify and warn of any erroneous attribute values.
pooling	true	(optional) If false, disables connection pooling.

Properties

Most properties of the connector-connection-pool element are the names of setter methods of the managedconnectionfactory-class element in the ra.xml file. Properties of the connector-connection-pool element override the ManagedConnectionFactory JavaBean configuration settings.

All but the last four properties in the following table are connector-connection-pool properties of jmsra, the resource adapter used to communicate with the Open Message Queue software. For a complete list of the available properties (called administered object attributes in the Message Queue software), see the Open Message Queue Administration Guide.

Changes to connector-connection-pool properties require a server restart.

Table C-37 connector-connection-pool Properties

Property	Default	Description
AddressList	none	Specifies a list of host/port combinations of the Message Queue software. For JMS resources of the Type jakarta.jms.TopicConnectionFactory or jakarta.jms.QueueConnectionFactory.

Property	Default	Description
ClientId	none	Specifies the JMS Client Identifier to be associated with a Connection created using the createTopicConnection method of the TopicConnectionFactory class. For JMS resources of the Type jakarta.jms.TopicConnectionFactory . Durable subscription names are unique and only valid within the scope of a client identifier. To create or reactivate a durable subscriber, the connection must have a valid client identifier. The JMS specification ensures that client identifiers are unique and that a given client identifier is allowed to be used by only one active connection at a time.
UserName	guest	Specifies the user name for connecting to the Message Queue software. For JMS resources of the Type jakarta.jms.TopicConnectionFactory or jakarta.jms.QueueConnectionFactory.
Password	guest	Specifies the password for connecting to the Message Queue software. For JMS resources of the Type jakarta.jms.TopicConnectionFactory or jakarta.jms.QueueConnectionFactory.
ReconnectAttempts	6	Specifies the number of attempts to connect (or reconnect) for each address in the imqAddressList before the client runtime moves on to try the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds).
ReconnectInterval	30000	Specifies the interval between reconnect attempts in milliseconds. This applies to attempts on each address in the imqAddressList and on successive addresses in the list. If too short, this time interval does not give a broker time to recover. If too long, the reconnect might represent an unacceptable delay.
ReconnectEnabled	false	If true, specifies that the client runtime attempts to reconnect to a message server (or the list of addresses in imqAddressList) when a connection is lost.
AddressListBehavior	priority	Specifies whether connection attempts are in the order of addresses in the imqAddressList attribute (priority) or in a random order (random). If many clients are attempting a connection using the same connection factory, use a random order to prevent them from all being connected to the same address.

Property	Default	Description
AddressListIterations	-1	Specifies the number of times the client runtime iterates through the imqAddressList in an effort to establish (or reestablish) a connection. A value of -1 indicates that the number of attempts is unlimited.

All JMS administered object resource properties that worked with version 7 of the Eclipse GlassFish are supported for backward compatibility.

connector-resource

Defines the connection factory object of a specific connection definition in a connector (resource adapter).

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the connector-resource element.

Table C-38 connector-resource Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the connector-resource element.

Table C-39 connector-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
pool-name	none	Specifies the name of the associated connector-connection-pool.

Attribute	Default	Description
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

consistency

Specifies container behavior in guaranteeing transactional consistency of the data in the bean.

Superelements

entity-mapping (sun-cmp-mappings.xml)

Subelements

The following table describes subelements for the consistency element.

Table C-40 consistency Subelements

Element	Required	Description
none	exactly one subelement is required	No consistency checking occurs.
check-modified-at-commit	exactly one subelement is required	Checks concurrent modification of fields in modified beans at commit time.
lock-when-loaded	exactly one subelement is required	Obtains an exclusive lock when the data is loaded.
check-all-at-commit	+	This element is not implemented. Do not use.
lock-when-modified	+	This element is not implemented. Do not use.

Element	Required	Description
check-version-of-accessed-instances	exactly one subelement is required	Checks the version column of the modified beans.

constraint-field

Specifies a cacheability constraint for the given url-pattern or servlet-name.

All constraint-field constraints must pass for a response to be cached. If there are value constraints, at least one of them must pass.

Superelements

cache-mapping (glassfish-web.xml)

Subelements

The following table describes subelements for the constraint-field element.

Table C-41 constraint-field Subelements

Element	Required	Description
constraint-field-value	zero or more	Contains a value to be matched to the input parameter value.

Attributes

The following table describes attributes for the constraint-field element.

Table C-42 constraint-field Attributes

Attribute	Default	Description
name	none	Specifies the input parameter name.
scope	request.parameter	(optional) Specifies the scope from which the input parameter is retrieved. Allowed values are context.attribute, request.header, request.parameter, request.cookie, request.attribute, and session.attribute.
cache-on-match	true	(optional) If true, caches the response if matching succeeds. Overrides the same attribute in a constraint-field-value subelement.
cache-on-match-failure	false	(optional) If true, caches the response if matching fails. Overrides the same attribute in a constraint-field-value subelement.

constraint-field-value

Specifies a value to be matched to the input parameter value. The matching is case sensitive. For example:

```
<value match-expr="in-range">1-60</value>
```

Superelements

constraint-field (glassfish-web.xml)

Subelements

none - contains data

Attributes

The following table describes attributes for the constraint-field-value element.

Table C-43 constraint-field-value Attributes

Attribute	Default	Description
match-expr	equals	(optional) Specifies the type of comparison performed with the value. Allowed values are equals, not-equals, greater, lesser, and in-range. If match-expr is greater or lesser, the value must be a number. If match-expr is in-range, the value must be of the form n1`-`n2, where n1 and n2 are numbers.
cache-on-match	true	(optional) If true, caches the response if matching succeeds.
cache-on-match-failure	false	(optional) If true, caches the response if matching fails.

context-root

Contains the web context root for the application or web application that was packaged as a WAR file. Overrides the corresponding element in the application.xml or web.xml file.

If the parent element is java-web-start-access, this element contains the context root for the Java Web Start enabled application client module. If none is specified, a default is generated; see java-web-start-access.

If you are setting up load balancing, web module context roots must be unique within a server

instance. See the Eclipse GlassFish High Availability Administration Guide for more information about load balancing.

Superelements

`web` (`glassfish-application.xml`), `glassfish-web-app` (`glassfish-web.xml`), `java-web-start-access` (`glassfish-application-client.xml`)

Subelements

none - contains data

`cookie-properties`

Specifies session cookie properties.

If cookie settings are defined declaratively in the `web.xml` file, the cookie properties defined here take precedence. If cookie settings are defined programmatically using `javax.servlet.SessionCookieConfig` methods, those cookie settings take precedence over the cookie properties defined here.

Superelements

`session-config` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `cookie-properties` element.

Table C-44 `cookie-properties` Subelements

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The following table describes properties for the `cookie-properties` element.

Table C-45 `cookie-properties` Properties

Property	Default	Description
`cookieName`	none	Specifies the cookie name.

Property	Default	Description
cookiePath	Context path at which the web module is installed.	Specifies the pathname that is set when the cookie is created. The browser sends the cookie if the pathname for the request contains this pathname. If set to / (slash), the browser sends cookies to all URLs served by Eclipse GlassFish. You can set the path to a narrower mapping to limit the request URLs to which the browser sends cookies.
cookieMaxAgeSeconds	none	Specifies the expiration time (in seconds) after which the browser expires the cookie. If this is unset, the cookie doesn't expire.
cookieDomain	(unset)	Specifies the domain for which the cookie is valid.
cookieComment	none	Specifies the comment that identifies the session tracking cookie in the cookie file.
cookieSecure	dynamic	Sets the Secure attribute of any JSESSIONID cookies associated with the web application. Allowed values are as follows: • true — Sets Secure to true. • false — Sets Secure to false. • dynamic — The JSESSIONID cookie inherits the Secure setting of the request that initiated the session. To set the Secure attribute of a JSESSIONIDSSO cookie, use the ssoCookieSecure virtual-server property. For details, see create-virtual-server(1).
cookieHttpOnly	none	Specifies that the cookie is marked HTTP only. Allowed values are true or false.

create-tables-at-deploy

Specifies whether database tables are created for beans that are automatically mapped by the EJB container. If true, creates tables in the database. If false (the default if this element is not present), does not create tables.

This element can be overridden during deployment. See "Generation Options for CMP" in Eclipse GlassFish Application Development Guide.

Superelements

cmp-resource (glassfish-ejb-jar.xml)

Subelements

none - contains data

custom-resource

Defines a custom resource, which specifies a custom server-wide resource object factory. Such object factories implement the javax.naming.spi.ObjectFactory interface.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the custom-resource element.

Table C-46 custom-resource Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the custom-resource element.

Table C-47 custom-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
res-type	none	Specifies the fully qualified type of the resource.
factory-class	none	Specifies the fully qualified name of the user-written factory class, which implements javax.naming.spi.ObjectFactory.

Attribute	Default	Description
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

database-vendor-name

Specifies the name of the database vendor for which tables can be created. Allowed values are javadb, db2, mssql, mysql, oracle, postgresql, pointbase, derby (also for CloudScape), and sybase, case-insensitive.

If no value is specified, a connection is made to the resource specified by the jndi-name subelement of the cmp-resource element, and the database vendor name is read. If the connection cannot be established, or if the value is not recognized, SQL-92 compliance is presumed.

This element can be overridden during deployment. See "Generation Options for CMP" in Eclipse GlassFish Application Development Guide.

Superelements

cmp-resource (glassfish-ejb-jar.xml)

Subelements

none - contains data

debugging-enabled

Specifies whether the debugging servlet is enabled for this web service endpoint. Allowed values are true (the default) and false.

Superelements

webservice-endpoint (glassfish-web.xml, glassfish-ejb-jar.xml)

Subelements

none - contains data

default

Specifies that a field belongs to the default hierarchical fetch group, and enables prefetching for a CMR field. To disable prefetching for specific query methods, use a `prefetch-disabled` element in the `glassfish-ejb-jar.xml` file.

Superelements

`fetched-with` (`sun-cmp-mappings.xml`)

Subelements

none - element is present or absent

default-helper

Passes property values to the built-in `default` `cache-helper` class.

Superelements

`cache` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `default-helper` element.

Table C-48 `default-helper` Subelements

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The following table describes properties for the `default-helper` element.

Table C-49 `default-helper` Properties

Property	Default	Description
cacheKeyGeneratorAttrName	Uses the built-in default cache-helper key generation, which concatenates the servlet path with key-field values, if any.	The caching engine looks in the ServletContext for an attribute with a name equal to the value specified for this property to determine whether a customized CacheKeyGenerator implementation is used. An application can provide a customized key generator rather than using the default helper. See "The CacheKeyGenerator Interface" in Eclipse GlassFish Application Development Guide.

default-resource-principal

Specifies the default principal (user) for the resource.

If this element is used in conjunction with a JMS Connection Factory resource, the name and password subelements must be valid entries in the Open Message Queue broker user repository. See "Configuring and Managing Security Services" in Open Message Queue Administration Guide for details.

Superelements

resource-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml); cmp-resource, mdb-connection-factory (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the default-resource-principal element.

Table C-50 default-resource-principal Subelements

Element	Required	Description
name	only one	Specifies the default resource principal name used to sign on to a resource manager.
password	only one	Specifies password of the default resource principal.

description

Specifies a text description of the containing element.

Superelements

property (with attributes), valve (glassfish-web.xml); activation-config, method (glassfish-ejb-jar.xml); target-server (sun-acc.xml); admin-object-resource, connector-connection-pool, connector-resource, custom-resource, external-jndi-resource, jdbc-connection-pool, jdbc-resource, mail-resource, property (with attributes), resource-adapter-config (glassfish-resources.xml)

Subelements

none - contains data

disable-nonportable-jndi-names

Because the EJB 3.1 specification defines portable EJB JNDI names, there is less need for Eclipse GlassFish specific JNDI names. By default, Eclipse GlassFish specific default JNDI names are applied automatically for backward compatibility. To disable Eclipse GlassFish specific JNDI names for an EJB module, set the value of this element to `true`. The default is `false`.

Superelements

`glassfish-ejb-jar` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

dispatcher

Specifies a comma-separated list of `RequestDispatcher` methods for which caching is enabled on the target resource. Valid values are `REQUEST`, `FORWARD`, `INCLUDE`, and `ERROR` . If this element is not specified, the default is `REQUEST`. See SRV.6.2.5 of the Servlet 2.4 specification for more information.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none - contains data

drop-tables-at-undeploy

Specifies whether database tables that were automatically created when the bean(s) were last deployed are dropped when the bean(s) are undeployed. If `true`, drops tables from the database. If `false` (the default if this element is not present), does not drop tables.

This element can be overridden during deployment. See "Generation Options for CMP" in Eclipse GlassFish Application Development Guide.

Superelements

`cmp-resource` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

ejb

Defines runtime properties for a single enterprise bean within the application. The subelements listed below apply to particular enterprise beans as follows:

- All types of beans: `ejb-name`, `ejb-ref`, `resource-ref`, `resource-env-ref`, `ior-security-config`, `gen-classes`, `jndi-name`, `use-thread-pool-id`, `message-destination-ref`, `pass-by-reference`, `service-ref`
- Stateless session beans: `bean-pool`, `webservice-endpoint`
- Stateful session beans: `bean-cache`, `webservice-endpoint`, `checkpoint-at-end-of-method`
- Entity beans: `commit-option`, `bean-cache`, `bean-pool`, `cmp`, `is-read-only-bean`, `refresh-period-in-seconds`, `flush-at-end-of-method`
- Message-driven beans: `mdb-resource-adapter`, `mdb-connection-factory`, `jms-durable-subscription-name`, `jms-max-messages-load`, `bean-pool`

Superelements

`enterprise-beans` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `ejb` element.

Table C-51 `ejb` Subelements

Element	Required	Description
`ejb-name`	only one	Matches the `ejb-name` in the corresponding `ejb-jar.xml` file.
`jndi-name`	zero or more	Specifies the absolute `jndi-name`.
`ejb-ref`	zero or more	Maps the absolute JNDI name to the `ejb-ref` element in the corresponding Jakarta EE XML file.
`resource-ref`	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Jakarta EE XML file.
`resource-env-ref`	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Jakarta EE XML file.
`service-ref`	zero or more	Specifies runtime settings for a web service reference.
`message-destination-ref`	zero or more	Specifies the name of a physical message destination.
`pass-by-reference`	zero or one	Specifies the passing method used by an enterprise bean calling a remote interface method in another bean that is colocated within the same process.

Element	Required	Description
`cmp`	zero or one	Specifies runtime information for a container-managed persistence (CMP) entity bean for EJB 1.1 and EJB 2.1 beans.
`principal`	zero or one	Specifies the principal (user) name in an enterprise bean that has the `run-as` role specified.
`mdb-connection-factory`	zero or one	Specifies the connection factory associated with a message-driven bean.
`jms-durable-subscription-name`	zero or one	Specifies the durable subscription associated with a message-driven bean.
`jms-max-messages-load`	zero or one	Specifies the maximum number of messages to load into a Java Message Service session at one time for a message-driven bean to serve. The default is 1.
`ior-security-config`	zero or one	Specifies the security information for the IOR.
`is-read-only-bean`	zero or one	Specifies that this entity bean is read-only.
`refresh-period-in-seconds`	zero or one	Specifies the rate at which a read-only-bean must be refreshed from the data source.
`commit-option`	zero or one	Has valid values of B or C. Default value is B.
`cmt-timeout-in-seconds`	zero or one	Overrides the Transaction Timeout setting of the Transaction Service for an individual bean.
`use-thread-pool-id`	zero or one	Specifies the thread pool from which threads are selected for remote invocations of this bean.
`gen-classes`	zero or one	Specifies all the generated class names for a bean.
`bean-pool`	zero or one	Specifies the bean pool properties. Used for stateless session beans, entity beans, and message-driven beans.
`bean-cache`	zero or one	Specifies the bean cache properties. Used only for stateful session beans and entity beans.
`mdb-resource-adapter`	zero or one	Specifies runtime configuration information for a message-driven bean.
`webservice-endpoint`	zero or more	Specifies information about a web service endpoint.
`flush-at-end-of-method`	zero or one	Specifies the methods that force a database flush after execution. Used for entity beans.
`checkpointed-methods`	zero or one	Deprecated. Supported for backward compatibility. Use `checkpoint-at-end-of-method` instead.

Element	Required	Description
checkpoint-at-end-of-method	zero or one	Specifies that the stateful session bean state is checkpointed, or persisted, after the specified methods are executed. The availability-enabled attribute must be set to true.
per-request-load-balancing	zero or one	Specifies the per-request load balancing behavior of EJB 2.x and 3.x remote client invocations on a stateless session bean.

Attributes

The following table describes attributes for the ejb element.

Table C-52 ejb Attributes

Attribute	Default	Description
availability-enabled	false	(optional) If set to true, and if availability is enabled in the EJB container, high-availability features apply to this bean if it is a stateful session bean.

Example

```
<ejb>
   <ejb-name>CustomerEJB</ejb-name>
   <jndi-name>customer</jndi-name>
   <resource-ref>
      <res-ref-name>jdbc/SimpleBank</res-ref-name>
      <jndi-name>jdbc/__default</jndi-name>
   </resource-ref>
   <is-read-only-bean>false</is-read-only-bean>
   <commit-option>B</commit-option>
   <bean-pool>
      <steady-pool-size>10</steady-pool-size>
      <resize-quantity>10</resize-quantity>
      <max-pool-size>100</max-pool-size>
      <pool-idle-timeout-in-seconds>600</pool-idle-timeout-in-seconds>
   </bean-pool>
   <bean-cache>
      <max-cache-size>100</max-cache-size>
      <resize-quantity>10</resize-quantity>
      <removal-timeout-in-seconds>3600</removal-timeout-in-seconds>
      <victim-selection-policy>LRU</victim-selection-policy>
   </bean-cache>
</ejb>
```

ejb-name

In the glassfish-ejb-jar.xml file, matches the ejb-name in the corresponding ejb-jar.xml file. The name must be unique among the names of the enterprise beans in the same EJB JAR file.

There is no architected relationship between the ejb-name in the deployment descriptor and the JNDI name that the deployer assigns to the EJB component's home.

In the sun-cmp-mappings.xml file, specifies the ejb-name of the entity bean in the ejb-jar.xml file to which the container-managed persistence (CMP) bean corresponds.

Superelements

ejb, method (glassfish-ejb-jar.xml); entity-mapping (sun-cmp-mappings.xml)

Subelements

none - contains data

ejb-ref

Maps the ejb-ref-name in the corresponding Jakarta EE deployment descriptor file ejb-ref entry to the absolute jndi-name of a resource.

The ejb-ref element is used for the declaration of a reference to an EJB's home. Applies to session beans or entity beans.

Superelements

glassfish-web-app (glassfish-web.xml), ejb (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the ejb-ref element.

Table C-53 ejb-ref Subelements

Element	Required	Description
ejb-ref-name	only one	Specifies the ejb-ref-name in the corresponding Jakarta EE deployment descriptor file ejb-ref entry.
jndi-name	only one	Specifies the absolute jndi-name of a resource.

ejb-ref-name

Specifies the ejb-ref-name in the corresponding Jakarta EE deployment descriptor file ejb-ref entry.

Superelements

`ejb-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

`eligible`

Specifies whether the application client module is eligible to be Java Web Start enabled. Allowed values are `true` (the default) and `false`.

Superelements

`java-web-start-access` (`glassfish-application-client.xml`)

Subelements

none - contains data

`endpoint-address-uri`

Specifies the relative path combined with the web server root to form the fully qualified endpoint address for a web service endpoint. This is a required element for EJB endpoints and an optional element for servlet endpoints.

For servlet endpoints, this value is relative to the web application context root. For EJB endpoints, the URI is relative to root of the web server (the first portion of the URI is a context root). The context root portion must not conflict with the context root of any web application deployed to the same web server.

In all cases, this value must be a fixed pattern (no "`*' allowed).

If the web service endpoint is a servlet that implements only a single endpoint and has only one `url-pattern`, it is not necessary to set this value, because the web container derives it from the `web.xml` file.

Superelements

`webservice-endpoint` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

Example

If the web server is listening at `http://localhost:8080`, the following `endpoint-address-uri`:

```
<endpoint-address-uri>StockQuoteService/StockQuotePort</endpoint-address-uri>
```

results in the following target endpoint address:

```
http://localhost:8080/StockQuoteService/StockQuotePort
```

enterprise-beans

Specifies all the runtime properties for an EJB JAR file in the application.

Superelements

glassfish-ejb-jar (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the enterprise-beans element.

Table C-54 enterprise-beans Subelements

Element	Required	Description
name	zero or one	Specifies the name string.
unique-id	zero or one	Specifies a unique system identifier. This data is automatically generated and updated at deployment/redeployment. Do not specify or edit this value.
ejb	zero or more	Defines runtime properties for a single enterprise bean within the application.
pm-descriptors	zero or one	Deprecated.
cmp-resource	zero or one	Specifies the database to be used for storing container-managed persistence (CMP) beans in an EJB JAR file.
message-destination	zero or more	Specifies the name of a logical message destination.
webservice-description	zero or more	Specifies a name and optional publish location for a web service.
property (with subelements)	zero or more	Specifies a property or a variable.

Example

```
<enterprise-beans>
```

```
  <ejb>
      <ejb-name>CustomerEJB</ejb-name>
      <jndi-name>customer</jndi-name>
      <resource-ref>
          <res-ref-name>jdbc/SimpleBank</res-ref-name>
          <jndi-name>jdbc/__default</jndi-name>
      </resource-ref>
      <is-read-only-bean>false</is-read-only-bean>
      <commit-option>B</commit-option>
      <bean-pool>
          <steady-pool-size>10</steady-pool-size>
         <resize-quantity>10</resize-quantity>
          <max-pool-size>100</max-pool-size>
          <pool-idle-timeout-in-seconds>600</pool-idle-timeout-in-seconds>
      </bean-pool>
      <bean-cache>
          <max-cache-size>100</max-cache-size>
          <resize-quantity>10</resize-quantity>
          <removal-timeout-in-seconds>3600</removal-timeout-in-seconds>
          <victim-selection-policy>LRU</victim-selection-policy>
      </bean-cache>
  </ejb>
 </enterprise-beans>
```

entity-mapping

Specifies the mapping a bean to database columns.

Superelements

sun-cmp-mapping (sun-cmp-mappings.xml)

Subelements

The following table describes subelements for the entity-mapping element.

Table C-55 entity-mapping Subelements

Element	Required	Description
ejb-name	only one	Specifies the name of the entity bean in the ejb-jar.xml file to which the CMP bean corresponds.
table-name	only one	Specifies the name of a database table. The table must be present in the database schema file.
cmp-field-mapping	one or more	Associates a field with one or more columns to which it maps.
cmr-field-mapping	zero or more	A container-managed relationship field has a name and one or more column pairs that define the relationship.

Element	Required	Description
secondary-table	zero or more	Describes the relationship between a bean's primary and secondary table.
consistency	zero or one	Specifies container behavior in guaranteeing transactional consistency of the data in the bean.

establish-trust-in-client

Specifies if the target is capable of authenticating a client. The values are NONE, SUPPORTED, or REQUIRED.

Superelements

transport-config (glassfish-ejb-jar.xml)

Subelements

none - contains data

establish-trust-in-target

Specifies if the target is capable of authenticating to a client. The values are NONE, SUPPORTED, or REQUIRED.

Superelements

transport-config (glassfish-ejb-jar.xml)

Subelements

none - contains data

external-jndi-resource

Defines a resource that resides in an external JNDI repository. For example, a generic Java object could be stored in an LDAP server. An external JNDI factory must implement the javax.naming.spi.InitialContextFactory interface.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the external-jndi-resource element.

Table C-56 external-jndi-resource Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the external-jndi-resource element.

Table C-57 external-jndi-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
jndi-lookup-name	none	Specifies the JNDI lookup name for the resource.
res-type	none	Specifies the fully qualified type of the resource.
factory-class	none	Specifies the fully qualified name of the factory class, which implements javax.naming.spi.InitialContextFactory . For more information about JNDI, see the Eclipse GlassFish Application Development Guide.
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

fetched-with

Specifies the fetch group configuration for fields and relationships. The `fetched-with` element has different allowed and default subelements based on its parent element and the data types of the fields.

- If there is no `fetched-with` subelement of a `cmp-field-mapping`, and the data type is not BLOB, CLOB, VARBINARY, LONGVARBINARY, or OTHER, `fetched-with` can have any valid subelement. The default subelement is as follows:

```
<fetched-with><default/></fetched-with>
```

- If there is no `fetched-with` subelement of a `cmp-field-mapping`, and the data type is BLOB, CLOB, VARBINARY, LONGVARBINARY, or OTHER, `fetched-with` can have any valid subelement except `<default/>`. The default subelement is as follows:

```
<fetched-with><none/></fetched-with>
```

- If there is no `fetched-with` subelement of a `cmr-field-mapping`, `fetched-with` can have any valid subelement. The default subelement is as follows:

```
<fetched-with><none/></fetched-with>
```

Managed fields are multiple CMP or CMR fields that are mapped to the same column. A managed field can have any `fetched-with` subelement except `<default/>`. For additional information, see "Managed Fields" in Eclipse GlassFish Application Development Guide.

Superelements

`cmp-field-mapping`, `cmr-field-mapping` (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `fetched-with` element.

Table C-58 `fetched-with` Subelements

Element	Required	Description
`default`	exactly one subelement is required	Specifies that a CMP field belongs to the default hierarchical fetch group, which means it is fetched any time the bean is loaded from a database. Enables prefetching of a CMR field.
`level`	exactly one subelement is required	Specifies the level number of a hierarchical fetch group.

Element	Required	Description
named-group	exactly one subelement is required	Specifies the name of an independent fetch group.
none	exactly one subelement is required	Specifies that this field or relationship is placed into its own individual fetch group, which means it is loaded from a database the first time it is accessed in this transaction.

field-name

Specifies the Java identifier of a field. This identifier must match the value of the field-name subelement of the cmp-field element in the ejb-jar.xml file.

Superelements

cmp-field-mapping (sun-cmp-mappings.xml)

Subelements

none - contains data

finder

Describes the finders for CMP 1.1 with a method name and query.

Superelements

one-one-finders (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the finder element.

Table C-59 finder Subelements

Element	Required	Description
method-name	only one	Specifies the method name for the finder.
query-params	zero or one	Specifies the query parameters for the CMP 1.1 finder.
query-filter	zero or one	Specifies the query filter for the CMP 1.1 finder.
query-variables	zero or one	Specifies variables in query expression for the CMP 1.1 finder.
query-ordering	zero or one	Specifies the query ordering for the CMP 1.1 finder.

flush-at-end-of-method

Specifies the methods that force a database flush after execution. Applicable to entity beans.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the flush-at-end-of-method element.

Table C-60 flush-at-end-of-method Subelements

Element	Required	Description
method	one or more	Specifies a bean method.

gen-classes

Specifies all the generated class names for a bean.

This value is automatically generated by the server at deployment or redeployment time. Do not specify it or change it after deployment.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the gen-classes element.

Table C-61 gen-classes Subelements

Element	Required	Description
remote-impl	zero or one	Specifies the fully-qualified class name of the generated EJBObject impl class.
local-impl	zero or one	Specifies the fully-qualified class name of the generated EJBLocalObject impl class.
remote-home-impl	zero or one	Specifies the fully-qualified class name of the generated EJBHome impl class.
local-home-impl	zero or one	Specifies the fully-qualified class name of the generated EJBLocalHome impl class.

glassfish-application

Defines the Eclipse GlassFish specific configuration for an application. This is the root element; there can only be one `glassfish-application` element in a `glassfish-application.xml` file. See The glassfish-application.xml File.

Superelements

none

Subelements

The following table describes subelements for the `glassfish-application` element.

Table C-62 `glassfish-application` Subelements

Element	Required	Description
`web`	zero or more	Specifies the application's web tier configuration.
`pass-by-reference`	zero or one	Determines whether EJB modules use pass-by-value or pass-by-reference semantics.
`unique-id`	zero or one	Contains the unique ID for the application.
`security-role-mapping`	zero or more	Maps a role in the corresponding Jakarta EE XML file to a user or group.
`realm`	zero or one	Specifies an authentication realm.
`ejb-ref`	zero or more	Maps the absolute JNDI name to the `ejb-ref` in the corresponding Jakarta EE XML file.
`resource-ref`	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Jakarta EE XML file.
`resource-env-ref`	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Jakarta EE XML file.
`service-ref`	zero or more	Specifies runtime settings for a web service reference.
`message-destination-ref`	zero or more	Specifies the name of a physical message destination.
`message-destination`	zero or more	Specifies the name of a logical message destination.
`archive-name`	zero or one	Specifies the name of the archive file.

Element	Required	Description
compatibility	zero or one	Specifies the Eclipse GlassFish release with which to be backward compatible in terms of JAR visibility requirements for applications.
keep-state	zero or one	Retains web sessions, stateful session bean instances, and persistently created EJB timers across redeployments.
version-identifier	zero or one	Contains version information for an application.

glassfish-application-client

Defines the Eclipse GlassFish specific configuration for an application client. This is the root element; there can only be one glassfish-application-client element in a glassfish-application-client.xml file. See The glassfish-application-client.xml file.

Superelements

none

Subelements

The following table describes subelements for the glassfish-application-client element.

Table C-63 glassfish-application-client subelements

Element	Required	Description
ejb-ref	zero or more	Maps the absolute JNDI name to the ejb-ref in the corresponding Jakarta EE XML file.
resource-ref	zero or more	Maps the absolute JNDI name to the resource-ref in the corresponding Jakarta EE XML file.
resource-env-ref	zero or more	Maps the absolute JNDI name to the resource-env-ref in the corresponding Jakarta EE XML file.
service-ref	zero or more	Specifies runtime settings for a web service reference.
message-destination-ref	zero or more	Specifies the name of a physical message destination.
message-destination	zero or more	Specifies the name of a logical message destination.
java-web-start-access	zero or one	Specifies changes to default Java Web Start parameters.
version-identifier	zero or one	Contains version information for an application client.

glassfish-ejb-jar

Defines the Eclipse GlassFish specific configuration for an EJB JAR file. This is the root element; there can only be one `glassfish-ejb-jar` element in a `glassfish-ejb-jar.xml` file. See The glassfish-ejb-jar.xml File.

Superelements

none

Subelements

The following table describes subelements for the `glassfish-ejb-jar` element.

Table C-64 `glassfish-ejb-jar` Subelements

Element	Required	Description
`security-role-mapping`	zero or more	Maps a role in the corresponding Jakarta EE XML file to a user or group.
`enterprise-beans`	only one	Describes all the runtime properties for an EJB JAR file in the application.
`compatibility`	zero or one	Specifies the Eclipse GlassFish release with which to be backward compatible in terms of JAR visibility requirements for applications.
`disable-nonportable-jndi-names`	zero or one	Disables Eclipse GlassFish specific JNDI names.
`keep-state`	zero or one	Retains stateful session bean instances and persistently created EJB timers across redeployments.
`version-identifier`	zero or one	Contains version information for an EJB module.

glassfish-web-app

Defines Eclipse GlassFish specific configuration for a web module. This is the root element; there can only be one `glassfish-web-app` element in a `glassfish-web.xml` file. See The glassfish-web.xml File.

Superelements

none

Subelements

The following table describes subelements for the `glassfish-web-app` element.

Table C-65 `glassfish-web-app` Subelements

Element	Required	Description
`context-root`	zero or one	Contains the web context root for the web module.
`security-role-mapping`	zero or more	Maps roles to users or groups in the currently active realm.
`servlet`	zero or more	Specifies a principal name for a servlet, which is used for the `run-as` role defined in `web.xml`.
`idempotent-url-pattern`	zero or more	Specifies a URL pattern for idempotent requests.
`session-config`	zero or one	Specifies session manager, session cookie, and other session-related information.
`ejb-ref`	zero or more	Maps the absolute JNDI name to the `ejb-ref` in the corresponding Jakarta EE XML file.
`resource-ref`	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Jakarta EE XML file.
`resource-env-ref`	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Jakarta EE XML file.
`service-ref`	zero or more	Specifies runtime settings for a web service reference.
`message-destination-ref`	zero or more	Specifies the name of a physical message destination.
`cache`	zero or one	Configures caching for web application components.
`class-loader`	zero or one	Specifies class loader configuration information.
`jsp-config`	zero or one	Specifies JSP configuration information.
`locale-charset-info`	zero or one	Deprecated. Use the `parameter-encoding` subelement of `glassfish-web-app` instead.
`parameter-encoding`	zero or one	Determines the default request character encoding and how the web container decodes parameters from forms according to a hidden field value.
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.
`valve`	zero or more	Specifies a custom valve.
`message-destination`	zero or more	Specifies the name of a logical message destination.

Element	Required	Description
webservice-description	zero or more	Specifies a name and optional publish location for a web service.
keep-state	zero or one	Retains web sessions across redeployments.
version-identifier	zero or one	Contains version information for a web application.

Attributes

The following table describes attributes for the glassfish-web-app element.

Table C-66 glassfish-web-app Attributes

Attribute	Default	Description
error-url	(blank)	(optional) Not implemented. Do not use.
httpservlet-security-provider	none	(optional) Specifies the HttpServlet message layer provider that the web container's servlet auth-constraint processing calls.

Properties

The following table describes properties for the glassfish-web-app element.

Table C-67 glassfish-web-app Properties

Property	Default	Description
allowLinking	false	If true, resources in this web application that are symbolic links are served. You can also define this property for a virtual server. Web applications on the virtual server that do not define this property use the virtual server's value. For details, see create-virtual-server(1). Caution: Setting this property to true on Windows systems exposes JSP source code.

Property	Default	Description
`alternatedocroot_n`	none	Specifies an alternate document root (docroot), where n is a positive integer that allows specification of more than one. Alternate docroots allow web applications to serve requests for certain resources from outside their own docroot, based on whether those requests match one (or more) of the URI patterns of the web application's alternate docroots. If a request matches an alternate docroot's URI pattern, it is mapped to the alternate docroot by appending the request URI (minus the web application's context root) to the alternate docroot's physical location (directory). If a request matches multiple URI patterns, the alternate docroot is determined according to the following precedence order: • Exact match • Longest path match • Extension match For example, the following properties specify three alternate docroots. The URI pattern of the first alternate docroot uses an exact match, whereas the URI patterns of the second and third alternate docroots use extension and longest path prefix matches, respectively.

Property	Default	Description
		<property name= "alternatedocroot_1" value="from=/my.jpg dir=/srv/images/jpg"/> <property name= "alternatedocroot_2" value="from=*.jpg dir=/srv/images/jpg"/> <property name= "alternatedocroot_3" value="from=/jpg/* dir=/src/images"/> The `value` of each alternate docroot has two components: The first component, `from`, specifies the alternate docroot's URI pattern, and the second component, `dir`, specifies the alternate docroot's physical location (directory). Spaces are allowed in the `dir` component. You can set this property for all the web applications on a specific virtual server. For details, see `create-virtual-server`(1).

Property	Default	Description
valve_n	none	This property is deprecated. Use the valve subelement instead. Specifies a fully qualified class name of a custom valve, where n is a positive integer that allows specification of more than one. The valve class must implement the org.apache.catalina.Valve interface from Tomcat or previous Eclipse GlassFish releases, or the org.glassfish.web.valve.GlassFishValve interface from the current Eclipse GlassFish release. For example: `<property name="valve_1" value= "org.glassfish.extension.Valve"/ >` You can set this property for all the web applications on a specific virtual server. For details, see create-virtual-server(1).
listener_n	none	Specifies a fully qualified class name of a custom Catalina listener, where n is a positive integer that allows specification of more than one. The listener class must implement the org.apache.catalina.ContainerListener, org.apache.catalina.LifecycleListener, or org.apache.catalina.InstanceListener interface. For example: `<property name="listener_1" value= "org.glassfish.extension.MyLifecycleListener"/>` You can set this property for all the web applications on a specific virtual server. For details, see create-virtual-server(1).
crossContextAllowed	true	If true, allows this web application to access the contexts of other web applications using the `ServletContext.``getContext()`method.

Property	Default	Description
relativeRedirectAllowed	false	If true, allows this web application to send a relative URL to the client using HttpServletResponse.``sendRedirect(), and instructs the web container not to translate any relative URLs to fully qualified ones.
reuseSessionID	false	If true, sessions generated for this web application use the session ID specified in the request.
securePagesWithPragma	true	Set this property to false to ensure that for this web application file downloads using SSL work properly in Internet Explorer. You can set this property for all the web applications on a specific virtual server. For details, see create-virtual-server(1).
singleThreadedServletPoolSize	5	Specifies the maximum number of servlet instances allocated for each SingleThreadModel servlet in the web application.
tempdir	domain-dir`/generated/`app-name or domain-dir`/generated/`module-name	Specifies a temporary directory for use by this web module. This value is used to construct the value of the javax.servlet.context.``tempdir context attribute. Compiled JSP files are also placed in this directory.
useResponseCTForHeaders	false	If true, response headers are encoded using the response's charset instead of the default (UTF-8).

group-map

Maps an EIS group to a group defined in the Eclipse GlassFish domain.

Superelements

work-security-map (glassfish-resources.xml)

Subelements

none

Attributes

The following table describes attributes for the `group-map` element.

Table C-68 `group-map` Attributes

Attribute	Default	Description
`eis-group`	none	Specifies an EIS group.
`mapped-group`	none	Specifies a group defined in the Eclipse GlassFish domain.

`group-name`

Specifies a group name in the current realm.

Superelements

`security-role-mapping` (`glassfish-application.xml`, `glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

`http-method`

Specifies an HTTP method that is eligible for caching. The default is `GET`.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none - contains data

`idempotent-url-pattern`

Specifies a URL pattern for idempotent requests.

Superelements

`glassfish-web-app` (`glassfish-web.xml`)

Subelements

none

Attributes

The following table describes attributes for the `idempotent-url-pattern` element.

Table C-69 `idempotent-url-pattern` Attributes

Attribute	Default	Description
`url-pattern`	none	Specifies a URL pattern, which can contain wildcards. The URL pattern must conform to the mappings specified in section SRV 11.2 of the Servlet 2.4 specification.
`no-of-retries`	`-1`	(optional) Specifies the number of times the load balancer retries an idempotent request. A value of `-1` indicates infinite retries.

Example

The following example specifies that all requests for the URI `sun-java/*` are idempotent.

```
<idempotent-url-pattern url-pattern="sun_java/*" no-of-retries="10"/>
```

integrity

Specifies if the target supports integrity-protected messages. The values are `NONE`, `SUPPORTED`, or `REQUIRED`.

Superelements

`transport-config` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

ior-security-config

Specifies the security information for the interoperable object reference (IOR).

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `ior-security-config` element.

Table C-70 `ior-security-config` Subelements

Element	Required	Description
`transport-config`	zero or one	Specifies the security information for transport.
`as-context`	zero or one	Specifies the authentication mechanism used to authenticate the client. If specified, it is `USERNAME_PASSWORD`.
`sas-context`	zero or one	Describes the sas-context fields.

`is-cache-overflow-allowed`

This element is deprecated. Do not use.

Superelements

`bean-cache` (`glassfish-ejb-jar.xml`)

`is-one-one-cmp`

This element is not used.

Superelements

`cmp` (`glassfish-ejb-jar.xml`)

`is-read-only-bean`

Specifies that this entity bean is a read-only bean if `true`. If this element is absent, the default value of `false` is used.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`java-method`

Specifies a method.

Superelements

`message` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

The following table describes subelements for the `java-method` element.

Table C-71 `java-method` Subelements

Element	Required	Description
`method-name`	only one	Specifies a method name.
`method-params`	zero or one	Specifies fully qualified Java type names of method parameters.

`java-web-start-access`

Specifies changes to default Java Web Start parameters for an embedded or stand-alone application client module.

Superelements

`glassfish-application-client` (`glassfish-application-client.xml`)

Subelements

The following table describes subelements for the `java-web-start-access` element.

Table C-72 `java-web-start-access` subelements

Element	Required	Description

`context-root`	zero or one	Contains the context root for the Java Web Start enabled application client module. If none is specified, a default is generated.
		
		The default for a web module is as follows:
		
		`http://host:port/app-name/relative-URI-to-appclient-jar`
		
		The default for a stand-alone application client module is as follows:
		
		`http://host:port/module-name`
		
		If the `module-name` is not specified during deployment, the name of the EAR or JAR file without the extension is used. If the web module is not in EAR or JAR file format, a name is generated and written to the server log.

`eligible`	zero or one	Specifies whether the application client module is eligible to be Java Web Start enabled. Allowed values are `true` (the default) and `false`.

Element	Required	Description
vendor	zero or one	Specifies the name of the vendor as it appears in Java Web Start download and launch screens. The default value is Application Client.
jnlp-doc	zero or one	Specifies the name of a custom JNLP file. If none is specified, a default JNLP file is generated.

jdbc-connection-pool

Defines the attributes and properties that are required for creating a JDBC connection pool.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the jdbc-connection-pool element.

Table C-73 jdbc-connection-pool Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the jdbc-connection-pool element. Changing the following attributes requires a server restart or the redeployment or disabling and re-enabling of applications that refer to the resource: datasource-classname, associate-with-thread, lazy-connection-association, and lazy-connection-enlistment.

Table C-74 jdbc-connection-pool Attributes

Attribute	Default	Description
name	none	Specifies the name of the connection pool. A jdbc-resource element's pool-name attribute refers to this name.

Attribute	Default	Description
`datasource-classname`	none	(optional) Specifies the class name of the associated vendor-supplied data source. This class must implement javax.sql.DataSource, javax.sql.XADataSource , javax.sql.ConnectionPool Datasource , or a combination.
`res-type`	none	(optional) Specifies the interface the data source class implements. The value of this attribute can be javax.sql.DataSource, javax.sql.XADataSource , javax.sql.ConnectionPool Datasource , or java.sql.Driver. To support configuration of JDBC drivers and applications that use java.sql.Driver implementations, set this attribute to java.sql.Driver . This attribute must be specified to avoid ambiguity when a data source class implements two or more of these interfaces or when a `driver-classname` is specified. An error occurs if this attribute has a legal value and the indicated interface is not implemented by the data source class.

Attribute	Default	Description
`driver-classname`	none	(optional) Specifies the vendor-supplied JDBC driver class name. This driver must implement the java.sql.Driver interface.
`ping`	`false`	(optional) Specifies whether to ping the pool during pool creation or reconfiguration to identify and warn of any erroneous attribute values.
`steady-pool-size`	`8`	(optional) Specifies the initial and minimum number of connections maintained in the pool.
`max-pool-size`	`32`	(optional) Specifies the maximum number of connections that can be created to satisfy client requests.
`max-wait-time-in-millis`	`60000`	(optional) Specifies the amount of time, in milliseconds, that the caller is willing to wait for a connection. If `0`, the caller is blocked indefinitely until a resource is available or an error occurs.

Attribute	Default	Description
pool-resize-quantity	2	(optional) Specifies the number of idle connections to be destroyed if the existing number of connections is above the steady-pool-size (subject to the max-pool-size limit). This is enforced periodically at the idle-timeout-in-seconds interval. An idle connection is one that has not been used for a period of idle-timeout-in-seconds. When the pool size reaches steady-pool-size, connection removal stops.
idle-timeout-in-seconds	300	(optional) Specifies the maximum time that a connection can remain idle in the pool. After this amount of time, the pool can close this connection. This timeout value must be kept shorter than the server side (database) timeout value to prevent the accumulation of unusable connections in the application.

Attribute	Default	Description
`transaction-isolation-level`	default JDBC driver isolation level	(optional) Specifies the transaction isolation level on the pooled database connections. Allowed values are `read-uncommitted`, `read-committed` , `repeatable-read`, or `serializable`. Applications that change the isolation level on a pooled connection programmatically risk polluting the pool, which can lead to errors. See `is-isolation-level-guaranteed` for more details.
`is-isolation-level-guaranteed`	`true`	(optional) Applicable only when `transaction-isolation-level` is explicitly set. If `true`, every connection obtained from the pool is guaranteed to have the desired isolation level. This might impact performance on some JDBC drivers. Only set this attribute to `false` if you are certain that the hosted applications do not return connections with altered isolation levels.
`is-connection-validation-required`	`false`	(optional) Specifies whether connections have to be validated before being given to the application. If a resource's validation fails, it is destroyed, and a new resource is created and returned.

Attribute	Default	Description
connection-validation-method	table	(optional) Legal values are as follows: • auto-commit, which uses Connection.setAutoCommit(Connection.``getAutoCommit()) • meta-data, which uses Connection.getMetaData() • table, which performs a query on a table specified in the validation-table-name attribute • custom-validation, which uses a user-defined validation mechanism specified by the custom implementation class in validation-classname. Because many JDBC drivers cache the results of auto-commit and meta-data calls, they do not always provide reliable validations. Check with the driver vendor to determine whether these calls are cached or not. The table must exist and be accessible, but it doesn't require any rows. Do not use an existing table that has a large number of rows or a table that is already frequently accessed.

Attribute	Default	Description
`validation-table-name`	none	(optional) Specifies the table name to be used to perform a query to validate a connection. This parameter is mandatory if and only if `connection-validation-method` is set to `table`.
`validation-classname`	none	(optional) Specifies the custom validation implementation class name. This parameter is mandatory if `connection-validation-method` is set to `custom-validation`. The classname provided must be accessible to the Eclipse GlassFish. The specified class must implement the org.glassfish.api.jdbc.ConnectionValidation interface. Eclipse GlassFish provides the following custom validation class templates for MSSQL, DB2, and Sybase databases. All of them implement the org.glassfish.api.jdbc.ConnectionValidation interface. • `org.glassfish.api.jdbc.MSSQLConnectionValidation` • `org.glassfish.api.jdbc.DB2ConnectionValidation` • `org.glassfish.api.jdbc.SybaseConnectionValidation`

Attribute	Default	Description
`init-sql`	none	(optional) Specifies an SQL string to be executed whenever a connection is created (not reused) in the pool. This initializes the state of the connection.
`fail-all-connections`	`false`	(optional) If `true`, closes all connections in the pool if a single validation check fails. This parameter is mandatory if and only if `is-connection-validation-required` is set to `true`.
`non-transactional-connections`	`false`	(optional) If `true`, non-transactional connections can be made to the JDBC connection pool. These connections are not automatically enlisted with the transaction manager.

Attribute	Default	Description
`allow-non-component-callers`	`false`	(optional) If `true`, non-Java-EE components, such as servlet filters, lifecycle modules, and third party persistence managers, can use this JDBC connection pool. The returned connection is automatically enlisted with the transaction context obtained from the transaction manager. Standard Jakarta EE components can also use such pools. Connections obtained by non-component callers are not automatically closed at the end of a transaction by the container. They must be explicitly closed by the caller.
`validate-atmost-once-period-in-seconds`	`0`	(optional) Specifies the time interval within which a connection is validated at most once. Minimizes the number of validation calls. A value of zero implies that Eclipse GlassFish does not attempt to minimize the number of validation requests by a connection. That is, a value of zero disables this attribute. As a result, the same connection is validated every time the application acquires the connection.

Attribute	Default	Description
`connection-leak-timeout-in-seconds`	`0`	(optional) Detects potential connection leaks by the application. A connection that is not returned back to the pool by the application within the specified period is assumed to be potentially leaking, and a stack trace of the caller is logged. A zero value disables leak detection. A nonzero value enables leak tracing. Use this attribute along with `connection-leak-reclaim` to avoid potential connection leaks from the application.
`connection-leak-reclaim`	`false`	(optional) If `true`, the pool will reclaim a connection after `connection-leak-timeout-in-seconds` occurs.
`connection-creation-retry-attempts`	`0`	(optional) Specifies the number of attempts to create a new connection in case of a failure.
`connection-creation-retry-interval-``in-seconds`	`10`	(optional) Specifies the time interval between attempts to create a connection when `connection-creation-retry-attempts` is greater than `0`.

Attribute	Default	Description
`statement-leak-timeout-in-seconds`	`0`	(optional) Detects potential statement leaks by the application. A statement that is not closed by the application within the specified period is assumed to be potentially leaking, and a stack trace of the caller is logged. A zero value disables leak detection. A nonzero value enables leak tracing. Use this attribute along with `statement-leak-reclaim` to avoid potential statement leaks from the application.
`statement-leak-reclaim`	`false`	(optional) If `true`, the reclaim of a statement after `statement-leak-timeout-in-seconds` occurs.

Attribute	Default	Description
statement-timeout-in-seconds	-1	(optional) Sets the query timeout property of a statement to enable termination of abnormally long running queries. The default value of -1 disables this feature. 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. This attribute 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 value specified in this attribute. This works only when the underlying JDBC driver supports queryTimeout for Statement, PreparedStatement, CallableStatement, and ResultSet.
lazy-connection-enlistment	false	(optional) If true, a connection is not enlisted in a transaction until it is used. If false, any connection object available to a transaction is enlisted in the transaction.

Attribute	Default	Description
`lazy-connection-association`	`false`	(optional) If `true`, a physical connection is not associated with a logical connection until it is used. If `false`, a physical connection is associated with a logical connection even before it is used.
`associate-with-thread`	`false`	(optional) Specifies whether connections are associated with the thread to enable the thread to reuse the connections. If `true`, allows connections to be saved as `ThreadLocal` in the calling thread. Connections get reclaimed only when the calling thread dies or when the calling thread is not in use and the pool has run out of connections. If `false`, the thread must obtain a connection from the pool each time the thread requires a connection.

Attribute	Default	Description
		This attribute associates connections with a thread such that when the same thread is in need of connections, it can reuse the connections already associated with that thread. In this case, the overhead of getting connections from the pool is avoided. However, when this value is set to `true`, you should verify that the value of the `max-pool-size` attribute is comparable to the `max-thread-pool-size` attribute of the associated thread pool. If the `max-thread-pool-size` value is much higher than the `max-pool-size` value, a lot of time is spent associating connections with a new thread after dissociating them from an older one. Use this attribute in cases where the thread pool should reuse connections to avoid this overhead.

Attribute	Default	Description
`match-connections`	`false`	(optional) Specifies whether a connection that is selected from the pool should be matched with the connections with certain credentials. If `true`, enables connection matching. You can set to `false` if connections are homogeneous. If the connection pool is used by applications that have multiple user credentials, `match-connections` must be `true`. The connection pool matches the request's credential with the connections in the pool and returns a matched connection for use. For new requests with different credentials, unmatched free connections are automatically purged to provide new connections to satisfy the new requests. This attribute need not be `true` if it is known that there is only one credential used by the applications and therefore the pool has homogeneous connections.

Attribute	Default	Description
`max-connection-usage-count`	`0`	(optional) Specifies the number of times a connections is reused by the pool, after which it is closed. A zero value disables this feature. By limiting the maximum number of times a connection can be reused, you can avoid statement leaks if the application does not close statements.
`sql-trace-listeners`	none	(optional) Specifies that SQL statements executed by applications need to be traced. Helps administrators analyze the statements. Expects as a value a comma-separated list of listener implementation class names. Enables easy filtering of log messages for the SQL statements. SQL trace listeners must implement the org.glassfish.api.jdbc.SQLTraceListener interface.
`statement-cache-size`	`0`	(optional) Specifies the number of statements to be cached using the `lru` (Least Recently Used) caching mechanism. The default value of `0` disables statement caching.
`pooling`	`true`	(optional) If `false`, disables connection pooling.

Attribute	Default	Description
wrap-jdbc-objects	true	(optional) If true, wrapped JDBC objects are returned for Statement, PreparedStatement, CallableStatement, ResultSet, and DatabaseMetaData. This option ensures that Statement.getConnection() is the same as DataSource.getConnection(). Therefore, this option should be true when both Statement.getConnection() and DataSource.getConnection() are done. The default is false to avoid breaking existing applications.

Eclipse GlassFish Properties

The following table describes properties for the jdbc-connection-pool element that are specific to Eclipse GlassFish.

Table C-75 jdbc-connection-pool Database Properties

Property	Default	Description
dynamic-reconfiguration-wait-timeout-in-seconds	none	Specifies the timeout for dynamic reconfiguration of the pool. In-progress connection requests must complete before this timeout expires or they must be retried. New connection requests wait for this timeout to expire before acquiring connections to the reconfigured pool. If this property exists and has a positive value, it is enabled. If this property is not set and pool reconfiguration results in pool recreation, in-progress connection requests must be retried.

Property	Default	Description
number-of-top-queries-to-report	10	Specifies the number of most frequently used queries to display. For example, the default value of 10 displays the top ten queries. This property is disabled when jdbc-connection-pool monitoring is set to LOW or OFF. It is enabled when jdbc-connection-pool monitoring is set to HIGH and the sql-trace-listeners attribute is set.
time-to-keep-queries-in-minutes	5	Specifies the time to retain queries in a cache before they are purged. This property is disabled when jdbc-connection-pool monitoring is set to LOW or OFF. It is enabled when jdbc-connection-pool monitoring is set to HIGH and the sql-trace-listeners attribute is set.

Database Properties

Most JDBC drivers allow use of standard property lists to specify the user, password, and other resource configuration information. Although properties are optional with respect to the Eclipse GlassFish, some properties might be necessary for most databases. For details, see the JDBC 4.0 Standard Extension API.

When properties are specified, they are passed to the vendor's data source class (specified by the datasource-classname attribute) as is using setName(value) methods.

The user and password properties are used as the default principal if container managed authentication is specified and a default-resource-principal is not found in the application deployment descriptors.

The following table describes some common properties for the jdbc-connection-pool element.

Changing JDBC driver properties requires a server restart.

Table C-76 jdbc-connection-pool Database Properties

Property	Description
user	Specifies the user name for connecting to the database.
password	Specifies the password for connecting to the database.

Property	Description
databaseName	Specifies the database for this connection pool.
serverName	Specifies the database server for this connection pool.
port	Specifies the port on which the database server listens for requests.
networkProtocol	Specifies the communication protocol.
roleName	Specifies the initial SQL role name.
datasourceName	Specifies an underlying XADataSource, or a ConnectionPoolDataSource if connection pooling is done.
description	Specifies a text description.
url	Specifies the URL for this connection pool. Although this is not a standard property, it is commonly used.

jdbc-resource

Defines a JDBC (javax.sql.DataSource) resource.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the jdbc-resource element.

Table C-77 jdbc-resource Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the jdbc-resource element.

Table C-78 jdbc-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
description	none	(optional) Specifies a text description of this element.

Attribute	Default	Description
pool-name	none	Specifies the name of the associated jdbc-connection-pool.
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

jms-durable-subscription-name

Specifies the durable subscription associated with a message-driven bean class. Only applies to the Java Message Service Topic Destination type, and only when the message-driven bean deployment descriptor subscription durability is Durable.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

none - contains data

jms-max-messages-load

Specifies the maximum number of messages to load into a Java Message Service session at one time for a message-driven bean to serve. The default is 1.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

none - contains data

jndi-name

Specifies the absolute `jndi-name` of a URL resource or a resource.

For entity beans and session beans, this value specifies the global JNDI name of the `EJBHome` object. It is only needed if the entity or session bean exposes a remote view.

For JMS message-driven beans, this is the JNDI name of the JMS resource from which the message-driven bean consumes JMS messages. This information is alternatively specified within the `activation-config` subelement of the `mdb-resource-adapter` element. For more information about JMS resources, see "Using the Java Message Service" in Eclipse GlassFish Application Development Guide.

Superelements

`ejb-ref`, `message-destination`, `resource-env-ref`, `resource-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`); `cmp-resource`, `ejb`, `mdb-connection-factory` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

jnlp-doc

Contains the name of a custom JNLP file, which modifies the behavior of a Java Web Start enabled application client module. If none is specified, a default JNLP file is generated.

The value of this element is a relative path with the following format:

```
[path-to-JAR-in-EAR!]path-to-JNLP-in-JAR
```

The default path-to-JAR-in-EAR is the current application client JAR file. For example, if the JNLP file is in the application client JAR file at `custom/myInfo.jnlp`, the element value would look like this:

```
<java-web-start-access>
   <jnlp-doc>custom/myInfo.jnlp</jnlp-doc>
</java-web-start-access>
```

If the application client is inside an EAR file, you can place the custom JNLP file inside another JAR file in the EAR. For example, if the JNLP file is in a JAR file at `other/myLib.jar`, the element value would look like this, with an exclamation point (`!`) separating the path to the JAR from the path in the JAR:

```
<java-web-start-access>
   <jnlp-doc>other/myLib.jar!custom/myInfo.jnlp</jnlp-doc>
```

```
</java-web-start-access>
```

For information about the allowed contents of a custom JNLP file, see "Developing Java Clients" in Eclipse GlassFish Application Development Guide.

Superelements

`java-web-start-access` (`glassfish-application-client.xml`)

Subelements

none - contains data

`jsp-config`

Specifies JSP configuration information.

Superelements

`glassfish-web-app` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `jsp-config` element.

Table C-79 `jsp-config` Subelements

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The default property values are tuned for development of JSP files at the cost of performance. To maximize performance, set `jsp-config` properties to these non-default values:

- `development` - `false` (as an alternative, set to `true` and give `modificationTestInterval` a large value)
- `mappedfile` - `false`
- `trimSpaces` - `true`
- `suppressSmap` - `true`
- `fork` - `false` (on Solaris)
- `classdebuginfo` - `false`

The following table describes properties for the `jsp-config` element.

Table C-80 `jsp-config` Properties

Property	Default	Description
`checkInterval`	`0`	If `development` is set to `false` and `checkInterval` is greater than zero, background compilations are enabled. The `checkInterval` is the time in seconds between checks to see if a JSP file needs to be recompiled.
`classdebuginfo`	`true`	Specifies whether the generated Java servlets are compiled with the debug option set (`-g` for `javac`).
`classpath`	created dynamically based on the current web application	Specifies the classpath to use when compiling generated servlets.
`compiler`	`javac`	Specifies the compiler Ant uses to compile JSP files. See the Ant documentation for more information: `http://antinstaller.sourceforge.net/manual/manual/`
`compilerSourceVM`	Depends on Eclipse GlassFish's Java runtime	Specifies the JDK release with which source compatibility of the generated servlets is provided. Same as the `-source` release option of `javac`. For more information, see `http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/javac.html#options`.
`compilerTargetVM`	Depends on Eclipse GlassFish's Java runtime	Specifies the Virtual Machine for the Java platform (JVM software) version for which the servlet class files are generated. Same as the `-target` release option of `javac`. For more information, see `http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/javac.html#options`.
`defaultBufferNone`	`false`	If `true`, the default for the `buffer` attribute of the `page` directive is `none`.
`development`	`true`	If set to `true`, enables development mode, which allows JSP files to be checked for modification. Specify the frequency at which JSPs are checked using the `modificationTestInterval` property.
`dumpSmap`	`false`	If set to `true`, dumps SMAP information for JSR 45 debugging to a file. Set to `false` if `suppressSmap` is `true`.
`enablePooling`	`true`	If set to `true`, tag handler pooling is enabled.

Property	Default	Description
enableTldValidation	false	If set to true, all Tag Library Descriptor (TLD) files referenced by the web application are validated against their underlying schema or DTD file.
errorOnUseBeanInvalidClassAttribute	false	If set to true, issues an error when the value of the class attribute in a useBean action is not a valid bean class.
fork	true	Specifies that Ant forks the compiling of JSP files, using a JVM machine separate from the one in which Tomcat is running.
genStrAsByteArray	true	If true, text strings are generated as bytes (encoded with the page encoding), if the page is not buffered.
genStrAsCharArray	false	If set to true, generates text strings as char arrays, which improves performance in some cases.
httpMethods	* for all methods	Specifies a comma separated list of HTTP methods supported by the JspServlet.
ignoreJspFragmentErrors	false	If set to true, instructs the compiler to ignore any JSP precompilation errors pertaining to statically included JSP segments that, despite not being top level JSP files, use the .jsp or .jspx extension (instead of the recommended .jspf).
initialCapacity	32	Specifies the initial capacity of the HashMap that maps JSP files to their corresponding servlets.
javaEncoding	UTF8	Specifies the encoding for the generated Java servlet. This encoding is passed to the Java compiler that is used to compile the servlet as well. By default, the web container tries to use UTF8. If that fails, it tries to use the javaEncoding value. For encodings, see: http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html
keepgenerated	true with JDK 5 and before and for jspc, otherwise false	If set to true, keeps the generated Java files. If false, deletes the Java files.
mappedfile	true	If set to true, generates static content with one print statement per input line, to ease debugging.

Property	Default	Description
modificationTestInterval	0	Specifies the frequency in seconds at which JSPs are checked for modification. A value of 0 causes the JSP to be checked on every access. Used only if development is set to true.
reload-interval	0	Specifies the frequency in seconds at which JSP files are checked for modifications. Setting this value to 0 checks JSP files for modifications on every request. Setting this value to -1 disables checks for JSP modifications and JSP recompilation.
saveBytecode	true for jspc, otherwise false	If true, generated byte code is saved to .class files? This option is meaningful only when the Java compiler API, JSR 199 (available with and used as the default on Java 6) is used for javac compilations.
scratchdir	The default work directory for the web applicatio n	Specifies the working directory created for storing all the generated code.
suppressSmap	false	If set to true, generation of SMAP information for JSR 45 debugging is suppressed.
trimSpaces	false	If set to true, trims white spaces in template text between actions or directives.
usePrecompiled	false	If set to true, an accessed JSP file is not compiled. Its precompiled servlet class is used instead. It is assumed that JSP files have been precompiled, and their corresponding servlet classes have been bundled in the web application's WEB-INF/lib or WEB-INF/classes directory.
xpoweredBy	true	If set to true, the X-Powered-By response header is added by the generated servlet.

keep-state

If set to true, retains web sessions, stateful session bean instances, and persistently created EJB timers across redeployments. The --keepstate option of the redeploy subcommand takes precedence. The default for both is false.

Some changes to an application between redeployments prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class.

For web applications, this feature is applicable only if in the `glassfish-web-app.xml` file the `persistence-type` attribute of the `session-manager` element is `file`.

For stateful session bean instances, the persistence type without high availability is set in the server (the `sfsb-persistence-type` attribute) and must be set to `file`, which is the default and recommended value.

If any active web session, SFSB instance, or EJB timer fails to be preserved or restored, none of these will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active state data, Eclipse GlassFish serializes the data and saves it in memory. To restore the data, the class loader of the newly redeployed application deserializes the data that was previously saved.

Superelements

`glassfish-application` (`glassfish-application.xml`), `glassfish-web-app` (`glassfish-web-app.xml`), `glassfish-ejb-jar` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

key-field

Specifies a component of the key used to look up and extract cache entries. The web container looks for the named parameter, or field, in the specified scope.

If this element is not present, the web container uses the Servlet Path (the path section that corresponds to the servlet mapping that activated the current request). See the Servlet 2.4 specification, section SRV 4.4, for details on the Servlet Path.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none

Attributes

The following table describes attributes for the `key-field` element.

Table C-81 `key-field` Attributes

Attribute	Default	Description
`name`	none	Specifies the input parameter name.

Attribute	Default	Description
scope	request.parameter	(optional) Specifies the scope from which the input parameter is retrieved. Allowed values are context.attribute, request.header, request.parameter, request.cookie, session.id, and session.attribute.

level

Specifies the name of a hierarchical fetch group. The name must be an integer. Fields and relationships that belong to a hierarchical fetch group of equal (or lesser) value are fetched at the same time. The value of level must be greater than zero. Only one is allowed.

Superelements

fetched-with (sun-cmp-mappings.xml)

Subelements

none - contains data

local-home-impl

Specifies the fully-qualified class name of the generated EJBLocalHome impl class.

This value is automatically generated by the server at deployment or redeployment time. Do not specify it or change it after deployment.

Superelements

gen-classes (glassfish-ejb-jar.xml)

Subelements

none - contains data

local-impl

Specifies the fully-qualified class name of the generated EJBLocalObject impl class.

This value is automatically generated by the server at deployment or redeployment time. Do not specify it or change it after deployment.

Superelements

gen-classes (glassfish-ejb-jar.xml)

Subelements

none - contains data

locale-charset-info

Deprecated. For backward compatibility only. Use the `parameter-encoding` subelement of `glassfish-web-app` instead. Specifies information about the application's internationalization settings.

Superelements

`glassfish-web-app` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `locale-charset-info` element.

Table C-82 `locale-charset-info` Subelements

Element	Required	Description
`locale-charset-map`	one or more	Maps a locale and an agent to a character encoding. Provided for backward compatibility. Used only for request processing, and only if no `parameter-encoding` is defined.
`parameter-encoding`	zero or one	Determines the default request character encoding and how the web container decodes parameters from forms according to a hidden field value.

Attributes

The following table describes attributes for the `locale-charset-info` element.

Table C-83 `locale-charset-info` Attributes

Attribute	Default	Description
`default-locale`	none	Although a value is required, the value is ignored. Use the `default-charset` attribute of the `parameter-encoding` element.

locale-charset-map

Maps locales and agents to character encodings. Provided for backward compatibility. Used only for request processing. Used only if the character encoding is not specified in the request and cannot be derived from the optional `parameter-encoding` element. For encodings, see `http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html`.

Superelements

`locale-charset-info` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `locale-charset-map` element.

Table C-84 `locale-charset-map` Subelements

Element	Required	Description
`description`	zero or one	Specifies an optional text description of a mapping.

Attributes

The following table describes attributes for the `locale-charset-map` element.

Table C-85 `locale-charset-map` Attributes

Attribute	Default	Description
`locale`	none	Specifies the locale name.
`agent`	none	(optional) Specifies the type of client that interacts with the Eclipse GlassFish. For a given locale, different agents can have different preferred character encodings. The value of this attribute must exactly match the value of the `user-agent` HTTP request header sent by the client. See Table C-86 for more information.
`charset`	none	Specifies the character encoding to which the locale maps.

Example Agents

The following table specifies example `agent` attribute values.

Table C-86 Example `agent` Attribute Values

Agent	`user-agent` Header and `agent` Attribute Value
Internet Explorer 5.00 for Windows 2000	`Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)`
Netscape 4.7.7 for Windows 2000	`Mozilla/4.77 [en] (Windows NT 5.0; U)`
Netscape 4.7 for Solaris	`Mozilla/4.7 [en] (X11; u; Sun OS 5.6 sun4u)`

`localpart`

Specifies the local part of a QNAME.

Superelements

`service-qname`, `wsdl-port` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

`lock-when-loaded`

Places a database update lock on the rows corresponding to the bean whenever the bean is loaded. How the lock is placed is database-dependent. The lock is released when the transaction finishes (commit or rollback). While the lock is in place, other database users have read access to the bean.

Superelements

`consistency` (`sun-cmp-mappings.xml`)

Subelements

none - element is present or absent

`lock-when-modified`

This element is not implemented. Do not use.

Superelements

`consistency` (`sun-cmp-mappings.xml`)

`log-service`

Specifies configuration settings for the log file.

Superelements

`client-container` (`sun-acc.xml`)

Subelements

The following table describes subelements for the `log-service` element.

Table C-87 `log-service` subelement

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the `log-service` element.

Table C-88 `log-service` attributes

Attribute	Default	Description
`log-file`	your-ACC-dir`/logs/client.log`	(optional) Specifies the file where the application client container logging information is stored.
`level`	`SEVERE`	(optional) Sets the base level of severity. Messages at or above this setting get logged to the log file.

`login-config`

Specifies the authentication configuration for an EJB web service endpoint. Not needed for servlet web service endpoints. A servlet's security configuration is contained in the `web.xml` file.

Superelements

`webservice-endpoint` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `login-config` element.

Table C-89 `login-config` subelements

Element	Required	Description
`auth-method`	only one	Specifies the authentication method.
`realm`	zero or one	Specifies the name of the realm used to process all authentication requests.

`mail-resource`

Defines a Jakarta Mail (`jakarta.mail.Session`) resource.

Superelements

`resources` (`glassfish-resources.xml`)

Subelements

The following table describes subelements for the `mail-resource` element.

Table C-90 `mail-resource` Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
property (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the mail-resource element.

Table C-91 mail-resource Attributes

Attribute	Default	Description
jndi-name	none	Specifies the JNDI name for the resource.
store-protocol	imap	(optional) Specifies the storage protocol service, which connects to a mail server, retrieves messages, and saves messages in folder(s). Allowed values are imap, pop3, imaps, and pop3s .
store-protocol-class	com.sun.mail.imap.IMAPStore	(optional) Specifies the service provider implementation class for storage. Allowed values are: com.sun.mail.imap.IMAPStore com.sun.mail.pop3.POP3Store com.sun.mail.imap.IMAPSSLStore com.sun.mail.pop3.POP3SSLStore
transport-protocol	smtp	(optional) Specifies the transport protocol service, which sends messages. Allowed values are smtp and smtps.
transport-protocol-class	com.sun.mail.smtp.SMTPTransport	(optional) Specifies the service provider implementation class for transport. Allowed values are: com.sun.mail.smtp.SMTPTransport com.sun.mail.smtp.SMTPSSLTransport
host	none	The mail server host name.
user	none	The mail server user name.
from	none	The email address the mail server uses to indicate the message sender.
debug	false	(optional) Determines whether debugging for this resource is enabled.

Attribute	Default	Description
object-type	user	(optional) Defines the type of the resource. Allowed values are: • system-all - A system resource for all server instances and the domain application server. • system-admin - A system resource only for the domain application server. • system-instance - A system resource for all server instances only. • user - A user resource.
enabled	true	(optional) Determines whether this resource is enabled at runtime.

Properties

You can set properties for the mail-resource element and then get these properties in a Jakarta Mail Session object later. Every property name must start with a mail- prefix. The Eclipse GlassFish changes the dash (-) character to a period (.) in the name of the property, then saves the property to the MailConfiguration and Jakarta Mail Session objects. If the name of the property doesn't start with mail-, the property is ignored.

For example, to define the property mail.password in a Jakarta Mail Session object, first edit glassfish-resources.xml as follows:

```
...
 <mail-resource jndi-name="mail/Session" ...>
     <property name="mail-password" value="adminadmin"/>
 </mail-resource>
 ...
```

After getting the Jakarta Mail Session object, get the mail.password property to retrieve the value adminadmin, as follows:

```
String password = session.getProperty("mail.password");
```

For more information about Jakarta Mail properties, see Jakarta Mail API Documentation (https://jakarta.ee/specifications/mail).

manager-properties

Specifies session manager properties.

Superelements

`session-manager` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `manager-properties` element.

Table C-92 `manager-properties` Subelements

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The following table describes properties for the `manager-properties` element.

Table C-93 `manager-properties` Properties

Property	Default	Description
`reapIntervalSeconds`	`60`	Specifies the number of seconds between checks for expired sessions. This is also the interval at which sessions are passivated if `maxSessions` is exceeded. If `persistenceFrequency` is set to `time-based`, active sessions are stored at this interval. To prevent data inconsistency, set this value lower than the frequency at which session data changes. For example, this value should be as low as possible (`1` second) for a hit counter servlet on a frequently accessed web site, or the last few hits might be lost each time the server is restarted. Applicable only if the `persistence-type` attribute of the parent `session-manager` element is `file` or `replicated`.
`maxSessions`	`-1`	Specifies the maximum number of sessions that are permitted in the cache, or `-1` for no limit. After this, an attempt to create a new session causes an `IllegalStateException` to be thrown. If the `persistence-type` attribute of the parent `session-manager` element is `file` or `replicated`, the session manager passivates sessions to the persistent store when this maximum is reached.

Property	Default	Description
sessionFilename	empty string	Specifies the absolute or relative path to the directory in which the session state is preserved between application restarts, if preserving the state is possible. A relative path is relative to the temporary directory for this web module, one of the following: domain-dir/generated/jsp/module-name domain-dir/generated/jsp/app-name`/``module-name By default, this property's value is set to an empty string, which disables this property and does not preserve the session state. Applicable only if the persistence-type attribute of the parent session-manager element is memory.
persistenceFrequency	web-method	Specifies how often the session state is stored. Allowed values are as follows: • web-method - The session state is stored at the end of each web request prior to sending a response back to the client. This mode provides the best guarantee that the session state is fully updated in case of failure. • time-based - The session state is stored in the background at the frequency set by reapIntervalSeconds. This mode provides less of a guarantee that the session state is fully updated. However, it can provide a significant performance improvement because the state is not stored after each request. Applicable only if the persistence-type attribute of the parent session-manager element is replicated.

mapping-properties

This element is not implemented.

Superelements

cmp (glassfish-ejb-jar.xml)

max-cache-size

Specifies the maximum number of beans allowable in cache. A value of zero indicates an unbounded cache. In reality, there is no hard limit. The max-cache-size limit is just a hint to the cache implementation. Default is 512.

Applies to stateful session beans and entity beans.

Superelements

`bean-cache` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`max-pool-size`

Specifies the maximum number of bean instances in the pool. Values are from 0 (1 for message-driven bean) to MAX_INTEGER. A value of 0 means the pool is unbounded. Default is 64.

Applies to all beans.

Superelements

`bean-pool` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`max-wait-time-in-millis`

This element is deprecated. Do not use.

Superelements

`bean-pool` (`glassfish-ejb-jar.xml`)

`mdb-connection-factory`

Specifies the connection factory associated with a message-driven bean. Queue or Topic type must be consistent with the Java Message Service Destination type associated with the message-driven bean class.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `mdb-connection-factory` element.

Table C-94 `mdb-connection-factory` Subelements

Element	Required	Description
jndi-name	only one	Specifies the absolute jndi-name.
default-resource-principal	zero or one	Specifies the default sign-on (name/password) to the resource manager.

mdb-resource-adapter

Specifies runtime configuration information for a message-driven bean.

Superelements

ejb (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the mdb-resource-adapter element.

Table C-95 mdb-resource-adapter subelements

Element	Required	Description
resource-adapter-mid	zero or one	Specifies a resource adapter module ID.
activation-config	one or more	Specifies an activation configuration.

message

Specifies the methods or operations to which message security requirements apply.

Superelements

message-security (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

The following table describes subelements for the message element.

Table C-96 message Subelements

Element	Required	Description
java-method	zero or one	Specifies the methods or operations to which message security requirements apply.
operation-name	zero or one	Specifies the WSDL name of an operation of a web service.

message-destination

Specifies the name of a logical message-destination defined within an application. The message-

destination-name matches the corresponding message-destination-name in the corresponding Jakarta EE deployment descriptor file. Use when the message destination reference in the corresponding Jakarta EE deployment descriptor file specifies a message-destination-link to a logical message-destination.

Superelements

glassfish-web-app (glassfish-web.xml), enterprise-beans (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the message-destination element.

Table C-97 message-destination subelements

Element	Required	Description
message-destination-name	only one	Specifies the name of a logical message destination defined within the corresponding Jakarta EE deployment descriptor file.
jndi-name	only one	Specifies the jndi-name of the associated entity.

message-destination-name

Specifies the name of a logical message destination defined within the corresponding Jakarta EE deployment descriptor file.

Superelements

message-destination (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

message-destination-ref

Directly binds a message destination reference to the JNDI name of a Queue, Topic, or other physical destination. Use only when the message destination reference in the corresponding Jakarta EE deployment descriptor file does not specify a message-destination-link to a logical message-destination.

Superelements

glassfish-web-app (glassfish-web.xml), ejb (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the `message-destination-ref` element.

Table C-98 `message-destination-ref` subelements

Element	Required	Description
`message-destination-ref-name`	only one	Specifies the name of a physical message destination defined within the corresponding Jakarta EE deployment descriptor file.
`jndi-name`	only one	Specifies the `jndi-name` of the associated entity.

`message-destination-ref-name`

Specifies the name of a physical message destination defined within the corresponding Jakarta EE deployment descriptor file.

Superelements

`message-destination-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

`message-security`

Specifies message security requirements.

- If the grandparent element is `webservice-endpoint`, these requirements pertain to request and response messages of the endpoint.
- If the grandparent element is `port-info`, these requirements pertain to the port of the referenced service.

Superelements

`message-security-binding` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

The following table describes subelements for the `message-security` element.

Table C-99 `message-security` Subelements

Element	Required	Description
message	one or more	Specifies the methods or operations to which message security requirements apply.
request-protection	zero or one	Defines the authentication policy requirements of the application's request processing.
response-protection	zero or one	Defines the authentication policy requirements of the application's response processing.

message-security-binding

Specifies a custom authentication provider binding for a parent webservice-endpoint or port-info element in one or both of these ways:

- By binding to a specific provider
- By specifying the message security requirements enforced by the provider

Superelements

webservice-endpoint, port-info (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

The following table describes subelements for the message-security-binding element.

Table C-100 message-security-binding Subelements

Element	Required	Description
message-security	zero or more	Specifies message security requirements.

Attributes

The following table describes attributes for the message-security-binding element.

Table C-101 message-security-binding Attributes

Attribute	Default	Description
auth-layer	none	Specifies the message layer at which authentication is performed. The value must be SOAP.

Attribute	Default	Description
provider-id	none	(optional) Specifies the authentication provider used to satisfy application-specific message security requirements. If this attribute is not specified, a default provider is used, if it is defined for the message layer. if no default provider is defined, authentication requirements defined in the message-security-binding are not enforced.

message-security-config

Specifies configurations for message security providers.

Superelements

client-container (sun-acc.xml)

Subelements

The following table describes subelements for the message-security-config element.

Table C-102 message-security-config Subelements

Element	Required	Description
provider-config	one or more	Specifies a configuration for one message security provider.

Attributes

The following table describes attributes for the message-security-config element.

Table C-103 message-security-config Attributes

Attribute	Default	Description
auth-layer	none	Specifies the message layer at which authentication is performed. The value must be SOAP.
default-provider	none	(optional) Specifies the server provider that is invoked for any application not bound to a specific server provider.

Attribute	Default	Description
default-client-provider	none	(optional) Specifies the client provider that is invoked for any application not bound to a specific client provider.

method

Specifies a bean method.

Superelements

checkpoint-at-end-of-method, flush-at-end-of-method (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the method element.

Table C-104 method Subelements

Element	Required	Description
description	zero or one	Specifies an optional text description.
ejb-name	zero or one	Matches the ejb-name in the corresponding ejb-jar.xml file.
method-name	only one	Specifies a method name.
method-intf	zero or one	Specifies the method interface to distinguish between methods with the same name in different interfaces.
method-params	zero or one	Specifies fully qualified Java type names of method parameters.

method-intf

Specifies the method interface to distinguish between methods with the same name in different interfaces. Allowed values are Home, Remote, LocalHome, and Local.

Superelements

method (glassfish-ejb-jar.xml)

Subelements

none - contains data

method-name

Specifies a method name or * (an asterisk) for all methods. If a method is overloaded, specifies all methods with the same name.

Superelements

java-method (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml); finder, query-method , method (glassfish-ejb-jar.xml)

Subelements

none - contains data

Examples

```
<method-name>findTeammates</method-name>
```

```
<method-name>*</method-name>
```

method-param

Specifies the fully qualified Java type name of a method parameter.

Superelements

method-params (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

method-params

Specifies fully qualified Java type names of method parameters.

Superelements

java-method (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml); query-method, method (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the method-params element.

Table C-105 method-params Subelements

Element	Required	Description
method-param	zero or more	Specifies the fully qualified Java type name of a method parameter.

name

Specifies the name of the entity.

Superelements

call-property, default-resource-principal, stub-property (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml); enterprise-beans, principal, property (with subelements) (glassfish-ejb-jar.xml)

Subelements

none - contains data

named-group

Specifies the name of one independent fetch group. All the fields and relationships that are part of a named group are fetched at the same time. A field belongs to only one fetch group, regardless of what type of fetch group is used.

Superelements

fetched-with (sun-cmp-mappings.xml)

Subelements

none - contains data

namespaceURI

Specifies the namespace URI.

Superelements

service-qname, wsdl-port (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

none

Specifies that this field or relationship is fetched by itself, with no other fields or relationships.

Superelements

consistency, fetched-with (sun-cmp-mappings.xml)

Subelements

none - element is present or absent

one-one-finders

Describes the finders for CMP 1.1 beans.

Superelements

cmp (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the one-one-finders element.

Table C-106 one-one-finders Subelements

Element	Required	Description
finder	one or more	Describes the finders for CMP 1.1 with a method name and query.

operation-name

Specifies the WSDL name of an operation of a web service.

Superelements

message (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

parameter-encoding

Specifies the default request character encoding and how the web container decodes parameters from forms according to a hidden field value.

If both the glassfish-web-app and locale-charset-info elements have parameter-encoding subelements, the subelement of glassfish-web-app takes precedence. For encodings, see http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html.

Superelements

`locale-charset-info`, `glassfish-web-app` (`glassfish-web.xml`)

Subelements

none

Attributes

The following table describes attributes for the `parameter-encoding` element.

Table C-107 `parameter-encoding` Attributes

Attribute	Default	Description
`form-hint-field`	none	(optional) The name of the hidden field in the form. This field specifies the character encoding the web container uses for `request.getParameter` and `request.getReader` calls when the charset is not set in the request's `content-type` header.
`default-charset`	`ISO-8859-1`	(optional) The default request character encoding.

`pass-by-reference`

Specifies the passing method used by a servlet or enterprise bean calling a remote interface method in another bean that is colocated within the same process.

- If `false` (the default if this element is not present), this application uses pass-by-value semantics.
- If `true`, this application uses pass-by-reference semantics.

The `pass-by-reference` element only applies to remote calls. As defined in the EJB 2.1 specification, section 5.4, calls to local interfaces use pass-by-reference semantics.

If the `pass-by-reference` element is set to its default value of `false`, the passing semantics for calls to remote interfaces comply with the EJB 2.1 specification, section 5.4. If set to `true`, remote calls involve pass-by-reference semantics instead of pass-by-value semantics, contrary to this specification.

Portable programs cannot assume that a copy of the object is made during such a call, and thus that it's safe to modify the original. Nor can they assume that a copy is not made, and thus that changes to the object are visible to both caller and callee. When this element is set to `true`, parameters and return values should be considered read-only. The behavior of a program that modifies such parameters or return values is undefined.

When a servlet or enterprise bean calls a remote interface method in another bean that is colocated within the same process, by default Eclipse GlassFish makes copies of all the call parameters in order to preserve the pass-by-value semantics. This increases the call overhead and decreases

performance.

However, if the calling method does not change the object being passed as a parameter, it is safe to pass the object itself without making a copy of it. To do this, set the pass-by-reference value to `true`.

The setting of this element in the `glassfish-application.xml` file applies to all EJB modules in the application. For an individually deployed EJB module, you can set the same element in the `glassfish-ejb-jar.xml` file. If `pass-by-reference` is used at both the bean and application level, the bean level takes precedence.

Superelements

`glassfish-application` (`glassfish-application.xml`), `ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

password

Specifies the password for the principal.

Superelements

`default-resource-principal` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

per-request-load-balancing

Specifies the per-request load balancing behavior of EJB 2.x and 3.x remote client invocations on a stateless session bean. If set to `true`, per-request load balancing is enabled for the associated stateless session bean. If set to `false` or not set, per-request load balancing is not enabled. The default is `false`.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

pm-descriptors

This element and its subelements are deprecated. Do not use.

Superelements

`enterprise-beans` (`glassfish-ejb-jar.xml`)

pool-idle-timeout-in-seconds

Specifies the maximum time, in seconds, that a bean instance is allowed to remain idle in the pool. When this timeout expires, the bean instance in a pool becomes a candidate for passivation or deletion. This is a hint to the server. A value of 0 specifies that idle beans remain in the pool indefinitely. Default value is 600.

Applies to stateless session beans, entity beans, and message-driven beans.

For a stateless session bean or a message-driven bean, the bean is removed (garbage collected) when the timeout expires.

Superelements

`bean-pool` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

port-component-name

Specifies a unique name for a port component within a web or EJB module.

Superelements

`webservice-endpoint` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

port-info

Specifies information for a port within a web service reference.

Either a `service-endpoint-interface` or a `wsdl-port` or both must be specified. If both are specified, `wsdl-port` specifies the port that the container chooses for container-managed port selection.

The same `wsdl-port` value must not appear in more than one `port-info` element within the same `service-ref`.

If a `service-endpoint-interface` is using container-managed port selection, its value must not appear in more than one `port-info` element within the same `service-ref`.

Superelements

`service-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

The following table describes subelements for the `port-info` element.

Table C-108 `port-info` subelements

Element	Required	Description
`service-endpoint-interface`	zero or one	Specifies the web service reference name relative to `java:comp/env`.
`wsdl-port`	zero or one	Specifies the WSDL port.
`stub-property`	zero or more	Specifies JAX-RPC property values that are set on a `javax.xml.rpc.Stub` object before it is returned to the web service client.
`call-property`	zero or more	Specifies JAX-RPC property values that are set on a `javax.xml.rpc.Call` object before it is returned to the web service client.
`message-security-binding`	zero or one	Specifies a custom authentication provider binding.

`prefetch-disabled`

Disables prefetching of entity bean states for the specified query methods. Container-managed relationship fields are prefetched if their `fetched-with` element is set to `default`.

Superelements

`cmp` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `prefetch-disabled` element.

Table C-109 `prefetch-disabled` Subelements

Element	Required	Description
`query-method`	one or more	Specifies a query method.

`principal`

Defines a user name on the platform.

Superelements

`ejb` (`glassfish-ejb-jar.xml`); `security-map` (`glassfish-resources.xml`)

Subelements

The following table describes subelements for the `principal` element.

Table C-110 `principal` Subelements

Element	Required	Description
`name`	only one	Specifies the name of the user.

`principal-map`

Maps an EIS principal to a principal defined in the Eclipse GlassFish domain.

Superelements

`work-security-map` (`glassfish-resources.xml`)

Subelements

none

Attributes

The following table describes attributes for the `principal-map` element.

Table C-111 `principal-map` Attributes

Attribute	Default	Description
`eis-principal`	none	Specifies an EIS principal.
`mapped-principal`	none	Specifies a principal defined in the Eclipse GlassFish domain.

`principal-name`

Contains the principal (user) name.

In an enterprise bean, specifies the principal (user) name that has the `run-as` role specified.

Superelements

`security-role-mapping` (`glassfish-application.xml`, `glassfish-web.xml`, `glassfish-ejb-jar.xml`), `servlet` (`glassfish-web.xml`)

Subelements

none - contains data

Attributes

The following table describes attributes for the `principal-name` element.

Table C-112 `principal-name` Attributes

Attribute	Default	Description
`class-name`	`com.sun.enterprise.deployment.UserNameAndPassword`	(optional) Specifies the custom principal implementation class corresponding to the named principal.

`property` (with attributes)

Specifies the name and value of a property. A property adds configuration information to its parent element that is one or both of the following:

- Optional with respect to Eclipse GlassFish
- Needed by a system or object that Eclipse GlassFish doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

`cache`, `cache-helper`, `class-loader`, `cookie-properties`, `default-helper`, `manager-properties`, `session-properties`, `store-properties`, `glassfish-web-app`, `valve`, `webservice-endpoint` (`glassfish-web.xml`); `auth-realm`, `client-container`, `client-credential`, `log-service`, `provider-config` (`sun-acc.xml`); `admin-object-resource`, `connector-connection-pool`, `connector-resource`, `custom-resource`, `external-jndi-resource`, `jdbc-connection-pool`, `jdbc-resource`, `mail-resource`, `resource-adapter-config` (`glassfish-resources.xml`)

Subelements

The following table describes subelements for the `property` element.

Table C-113 `property` Subelements

Element	Required	Description
`description`	zero or one	Specifies an optional text description of a property.

The `property` element in the `sun-acc.xml` file has no subelements.

Attributes

The following table describes attributes for the `property` element.

Table C-114 `property` Attributes

Attribute	Default	Description
`name`	none	Specifies the name of the property.
`value`	none	Specifies the value of the property.

Example

```
<property name="reapIntervalSeconds" value="20" />
```

`property` (with subelements)

Specifies the name and value of a property. A property adds configuration information to its parent element that is one or both of the following:

- Optional with respect to Eclipse GlassFish
- Needed by a system or object that Eclipse GlassFish doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

`enterprise-beans`, `cmp-resource`, `schema-generator-properties`, `webservice-endpoint` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `property` element.

Table C-115 `property` subelements

Element	Required	Description
`name`	only one	Specifies the name of the property.
`value`	only one	Specifies the value of the property.

Example

```
<property>
   <name>use-unique-table-names</name>
   <value>true</value>
</property>
```

`provider-config`

Specifies a configuration for one message security provider.

Although the request-policy and response-policy subelements are optional, the provider-config element does nothing if they are not specified.

Use property subelements to configure provider-specific properties. Property values are passed to the provider when its initialize method is called.

Superelements

message-security-config (sun-acc.xml)

Subelements

The following table describes subelements for the provider-config element.

Table C-116 provider-config Subelements

Element	Required	Description
request-policy	zero or one	Defines the authentication policy requirements of the authentication provider's request processing.
response-policy	zero or one	Defines the authentication policy requirements of the authentication provider's response processing.
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the provider-config element.

Table C-117 provider-config Attributes

Attribute	Default	Description
provider-id	none	Specifies the provider ID.
provider-type	none	Specifies whether the provider is a client, server, or client-server authentication provider.

Attribute	Default	Description
class-name	none	Specifies the Java implementation class of the provider. Client authentication providers must implement the com.sun.enterprise.security.jauth.ClientAuthModule interface. Server authentication providers must implement the com.sun.enterprise.security.jauth.ServerAuthModule interface. Client-server providers must implement both interfaces.

query-filter

Specifies the query filter for the CMP 1.1 finder.

Superelements

finder (glassfish-ejb-jar.xml)

Subelements

none - contains data

query-method

Specifies a query method.

Superelements

prefetch-disabled (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the query-method element.

Table C-118 query-method Subelements

Element	Required	Description
method-name	only one	Specifies a method name.
method-params	only one	Specifies the fully qualified Java type names of method parameters.

query-ordering

Specifies the query ordering for the CMP 1.1 finder.

Superelements

`finder` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`query-params`

Specifies the query parameters for the CMP 1.1 finder.

Superelements

`finder` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`query-variables`

Specifies variables in the query expression for the CMP 1.1 finder.

Superelements

`finder` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`read-only`

Specifies that a field is read-only if `true`. If this element is absent, the default value is `false` .

Superelements

`cmp-field-mapping` (`sun-cmp-mappings.xml`)

Subelements

none - contains data

`realm`

Specifies the name of the realm used to process all authentication requests associated with this application. If this element is not specified or does not match the name of a configured realm, the default realm is used. For more information about realms, see "Realm Configuration" in Eclipse

GlassFish Application Development Guide.

Superelements

`glassfish-application` (`glassfish-application.xml`), `as-context`, `login-config` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

refresh-field

Specifies a field that gives the application component a programmatic way to refresh a cached entry.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none

Attributes

The following table describes attributes for the `refresh-field` element.

Table C-119 `refresh-field` Attributes

Attribute	Default	Description
`name`	none	Specifies the input parameter name.
`scope`	`request.parameter`	(optional) Specifies the scope from which the input parameter is retrieved. Allowed values are `context.attribute`, `request.header`, `request.parameter`, `request.cookie`, `session.id`, and `session.attribute`.

refresh-period-in-seconds

Specifies the rate at which a read-only-bean must be refreshed from the data source. If the value is less than or equal to zero, the bean is never refreshed; if the value is greater than zero, the bean instances are refreshed at the specified interval. This rate is just a hint to the container. Default is 0 (no refresh).

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

removal-timeout-in-seconds

Specifies the amount of time a bean instance can remain idle in the container before it is removed (timeout). A value of 0 specifies that the container does not remove inactive beans automatically. The default value is 5400.

If removal-timeout-in-seconds is less than or equal to cache-idle-timeout-in-seconds, beans are removed immediately without being passivated.

Applies to stateful session beans.

For related information, see cache-idle-timeout-in-seconds.

Superelements

bean-cache (glassfish-ejb-jar.xml)

Subelements

none - contains data

remote-home-impl

Specifies the fully-qualified class name of the generated EJBHome impl class.

This value is automatically generated by the server at deployment or redeployment time. Do not specify it or change it after deployment.

Superelements

gen-classes (glassfish-ejb-jar.xml)

Subelements

none - contains data

remote-impl

Specifies the fully-qualified class name of the generated EJBObject impl class.

This value is automatically generated by the server at deployment or redeployment time. Do not specify it or change it after deployment.

Superelements

`gen-classes` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`request-policy`

Defines the authentication policy requirements of the authentication provider's request processing.

Superelements

`provider-config` (`sun-acc.xml`)

Subelements

none

Attributes

The following table describes attributes for the `request-policy` element.

Table C-120 `request-policy` Attributes

Attribute	Default	Description
`auth-source`	none	Specifies the type of required authentication, either `sender` (user name and password) or `content` (digital signature).
`auth-recipient`	none	Specifies whether recipient authentication occurs before or after content authentication. Allowed values are `before-content` and `after-content`.

`request-protection`

Defines the authentication policy requirements of the application's request processing.

Superelements

`message-security` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none

Attributes

The following table describes attributes for the `request-protection` element.

Table C-121 `request-protection` Attributes

Attribute	Default	Description
`auth-source`	none	Specifies the type of required authentication, either `sender` (user name and password) or `content` (digital signature).
`auth-recipient`	none	Specifies whether recipient authentication occurs before or after content authentication. Allowed values are `before-content` and `after-content`.

`required`

Specifies whether the authentication method specified in the `auth-method` element must be used for client authentication. The value is `true` or `false` (the default).

Superelements

`as-context` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`res-ref-name`

Specifies the `res-ref-name` in the corresponding Jakarta EE deployment descriptor file `resource-ref` entry. The `res-ref-name` element specifies the name of a resource manager connection factory reference. The name must be unique within an enterprise bean.

Superelements

`resource-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

`resize-quantity`

Specifies the number of bean instances to be:

- Created, if a request arrives when the pool has less than `steady-pool-size` quantity of beans

(applies to pools only for creation). If the pool has more than `steady-pool-size` minus `resize-quantity` of beans, then `resize-quantity` is still created.

- Removed, when the `pool-idle-timeout-in-seconds` timer expires and a cleaner thread removes any unused instances.
  - For caches, when `max-cache-size` is reached, `resize-quantity` beans are selected for passivation using the `victim-selection-policy`. In addition, the `cache-idle-timeout-in-seconds` or `removal-timeout-in-seconds` timers passivate beans from the cache.
  - For pools, when the `max-pool-size` is reached, `resize-quantity` beans are selected for removal. In addition, the `pool-idle-timeout-in-seconds` timer removes beans until `steady-pool-size` is reached.

Values are from 0 to MAX_INTEGER. The pool is not resized below the `steady-pool-size`. Default is 16.

Applies to stateless session beans, entity beans, and message-driven beans.

For EJB pools, the value can be defined in the EJB container. Default is 16. For EJB caches, the value can be defined in the EJB container. Default is 32.

For message-driven beans, the value can be defined in the EJB container. Default is 2.

Superelements

`bean-cache`, `bean-pool` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`resource-adapter-config`

Defines a connector (resource adapter) configuration. Stores configuration information for the resource adapter JavaBean in `property` subelements.

Superelements

`resources` (`glassfish-resources.xml`)

Subelements

The following table describes subelements for the `resource-adapter-config` element.

Table C-122 `resource-adapter-config` Subelements

Element	Required	Description
`property` (with attributes)	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the `resource-adapter-config` element.

Table C-123 `resource-adapter-config` Attributes

Attribute	Default	Description
`name`	none	(optional) Not used. See `resource-adapter-name`.
`thread-pool-ids`	none	(optional) Specifies a comma-separated list of the names of thread pools.
`object-type`	`user`	(optional) Defines the type of the resource. Allowed values are: • `system-all` - A system resource for all server instances and the domain application server. • `system-admin` - A system resource only for the domain application server. • `system-instance` - A system resource for all server instances only. • `user` - A user resource.
`resource-adapter-name`	none	Specifies the name of a deployed connector module or application. If the resource adapter is embedded in an application, then it is app_name`#`rar_name.

Properties

Properties of the `resource-adapter-config` element are the names of setter methods of the `resourceadapter-class` element in the `ra.xml` file, which defines the class name of the resource adapter JavaBean. Any properties defined here override the default values present in `ra.xml`.

`resource-adapter-mid`

Specifies the module ID of the resource adapter that is responsible for delivering messages to the message-driven bean.

Superelements

`mdb-resource-adapter` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

resource-env-ref

Maps the res-ref-name in the corresponding Jakarta EE deployment descriptor file resource-env-ref entry to the absolute jndi-name of a resource.

Superelements

glassfish-web-app (glassfish-web.xml), ejb (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the resource-env-ref element.

Table C-124 resource-env-ref Subelements

Element	Required	Description
resource-env-ref-name	only one	Specifies the res-ref-name in the corresponding Jakarta EE deployment descriptor file resource-env-ref entry.
jndi-name	only one	Specifies the absolute jndi-name of a resource.

Example

```
<resource-env-ref>
   <resource-env-ref-name>jms/StockQueueName</resource-env-ref-name>
   <jndi-name>jms/StockQueue</jndi-name>
</resource-env-ref>
```

resource-env-ref-name

Specifies the res-ref-name in the corresponding Jakarta EE deployment descriptor file resource-env-ref entry.

Superelements

resource-env-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

resource-ref

Maps the res-ref-name in the corresponding Jakarta EE deployment descriptor file resource-ref entry to the absolute jndi-name of a resource.

Connections acquired from JMS connection factories are not shareable in the current release of Eclipse GlassFish. The res-sharing-scope element in the ejb-jar.xml file resource-ref element is ignored for JMS connection factories.

When resource-ref specifies a JMS connection factory for the Open Message Queue, the default-resource-principal (name/password) must exist in the Message Queue user repository. Refer to the Security Management chapter in the Open Message Queue Administration Guide for information on how to manage the Message Queue user repository.

Superelements

glassfish-web-app (glassfish-web.xml), ejb (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the resource-ref element.

Table C-125 resource-ref Subelements

Element	Required	Description
res-ref-name	only one	Specifies the res-ref-name in the corresponding Jakarta EE deployment descriptor file resource-ref entry.
jndi-name	only one	Specifies the absolute jndi-name of a resource.
default-resource-principal	zero or one	Specifies the default principal (user) for the resource.

Example

```
<resource-ref>
   <res-ref-name>jdbc/EmployeeDBName</res-ref-name>
   <jndi-name>jdbc/EmployeeDB</jndi-name>
</resource-ref>
```

resources

Defines application-scoped resources for an enterprise application, web module, EJB module, connector module, or application client module. This is the root element; there can only be one resources element in a glassfish-resources.xml file. See The glassfish-resources.xml File.

You must specify a Java Naming and Directory Interface (JNDI) name for each resource. To avoid collisions with names of other enterprise resources in JNDI, and to avoid portability problems, all names in a Eclipse GlassFish application should begin with the string `java:app/`.

Superelements

none

Subelements

The following table describes subelements for the `resources` element.

Table C-126 `resources` Subelements

Element	Required	Description
`custom-resource`	zero or more	Defines a custom resource.
`external-jndi-resource`	zero or more	Defines a resource that resides in an external JNDI repository.
`jdbc-resource`	zero or more	Defines a JDBC (Java Database Connectivity) resource.
`mail-resource`	zero or more	Defines a Jakarta Mail resource.
`admin-object-resource`	zero or more	Defines an administered object for an inbound resource adapter.
`connector-resource`	zero or more	Defines a connector (resource adapter) resource.
`resource-adapter-config`	zero or more	Defines a resource adapter configuration.
`jdbc-connection-pool`	zero or more	Defines the properties that are required for creating a JDBC connection pool.
`connector-connection-pool`	zero or more	Defines the properties that are required for creating a connector connection pool.
`work-security-map`	zero or more	Defines a work security map.

Subelements of a `resources` element can occur in any order.

`response-policy`

Defines the authentication policy requirements of the authentication provider's response processing.

Superelements

`provider-config` (`sun-acc.xml`)

Subelements

none

Attributes

The following table describes attributes for the `response-policy` element.

Table C-127 `response-policy` Attributes

Attribute	Default	Description
`auth-source`	none	Specifies the type of required authentication, either `sender` (user name and password) or `content` (digital signature).
`auth-recipient`	none	Specifies whether recipient authentication occurs before or after content authentication. Allowed values are `before-content` and `after-content`.

`response-protection`

Defines the authentication policy requirements of the application's response processing.

Superelements

`message-security` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none

Attributes

The following table describes attributes for the `response-protection` element.

Table C-128 `response-protection` Attributes

Attribute	Default	Description
`auth-source`	none	Specifies the type of required authentication, either `sender` (user name and password) or `content` (digital signature).

Attribute	Default	Description
auth-recipient	none	Specifies whether recipient authentication occurs before or after content authentication. Allowed values are before-content and after-content.

role-name

Contains the role-name in the security-role element of the corresponding Jakarta EE deployment descriptor file.

Superelements

security-role-mapping (glassfish-application.xml, glassfish-web.xml, glassfish-ejb-jar.xml)

Subelements

none - contains data

sas-context

Describes the sas-context fields.

Superelements

ior-security-config (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the sas-context element.

Table C-129 sas-context Subelements

Element	Required	Description
caller-propagation	only one	Specifies whether the target accepts propagated caller identities. The values are NONE, SUPPORTED, or REQUIRED.

schema

Specifies the file that contains a description of the database schema to which the beans in this sun-cmp-mappings.xml file are mapped. If this element is empty, the database schema file is automatically generated at deployment time. Otherwise, the schema element names a .dbschema file with a pathname relative to the directory containing the sun-cmp-mappings.xml file, but without the .dbschema extension. See "Automatic Database Schema Capture" in Eclipse GlassFish Application Development Guide.

Superelements

`sun-cmp-mapping` (`sun-cmp-mappings.xml`)

Subelements

none - contains data

Examples

```
<schema/> <!-- use automatic schema generation -->

<schema>CompanySchema</schema> <!-- use "CompanySchema.dbschema" -->
```

schema-generator-properties

Specifies field-specific column attributes in `property` subelements.

Superelements

`cmp-resource` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `schema-generator-properties` element.

Table C-130 `schema-generator-properties` Subelements

Element	Required	Description
`property` (with subelements)	zero or more	Specifies a property name and value.

Properties

The following table describes properties for the `schema-generator-properties` element.

Table C-131 `schema-generator-properties` Properties

Property	Default	Description
`use-unique-table-names`	`false`	Specifies that generated table names are unique within each Eclipse GlassFish domain. This property can be overridden during deployment. See "Generation Options for CMP" in Eclipse GlassFish Application Development Guide.

Property	Default	Description
bean-name`.field-name.`attribute	none	Defines a column attribute. For attribute descriptions, see Table C-132.

The following table lists the column attributes for properties defined in the `schema-generator-properties` element.

Table C-132 `schema-generator-properties` Column Attributes

Attribute	Description
`jdbc-type`	Specifies the JDBC type of the column created for the CMP field. The actual SQL type generated is based on this JDBC type but is database vendor specific.
`jdbc-maximum-length`	Specifies the maximum number of characters stored in the column corresponding to the CMP field. Applies only when the actual SQL that is generated for the column requires a length. For example, a `jdbc-maximum-length` of `32` on a CMP `String` field such as `firstName` normally results in a column definition such as VARCHAR(32). But if the `jdbc-type` is `CLOB` and you are deploying on Oracle, the resulting column definition is CLOB. No length is given, because in an Oracle database, a CLOB has no length.
`jdbc-precision`	Specifies the maximum number of digits stored in a column which represents a numeric type.
`jdbc-scale`	Specifies the number of digits stored to the right of the decimal point in a column that represents a floating point number.
`jdbc-nullable`	Specifies whether the column generated for the CMP field allows null values.

Example

```
<schema-generator-properties>
   <property>
         <name>Employee.firstName.jdbc-type</name>
         <value>char</value>
   </property>
   <property>
         <name>Employee.firstName.jdbc-maximum-length</name>
         <value>25</value>
   </property>
   <property>
         <name>use-unique-table-names</name>
         <value>true</value>
   </property>
```

```
</schema-generator-properties>
```

secondary-table

Specifies a bean's secondary table(s).

Superelements

`entity-mapping` (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `secondary-table` element.

Table C-133 `secondary table` Subelements

Element	Required	Description
`table-name`	only one	Specifies the name of a database table.
`column-pair`	one or more	Specifies the pair of columns that determine the relationship between two database tables.

security

Defines the SSL security configuration for IIOP/SSL communication with the target server.

Superelements

`target-server` (`sun-acc.xml`)

Subelements

The following table describes subelements for the `security` element.

Table C-134 `security` Subelements

Element	Required	Description
`ssl`	only one	Specifies the SSL processing parameters.
`cert-db`	only one	Not implemented. Included for backward compatibility only.

security-map

Maps the principal received during servlet or EJB authentication to the credentials accepted by the EIS. This mapping is optional. It is possible to map multiple Eclipse GlassFish principals to the same back-end principal.

This is different from a `work-security-map`, which maps a principal associated with an incoming work instance to a principal in the Eclipse GlassFish's security domain.

Superelements

`connector-connection-pool` (`glassfish-resources.xml`)

Subelements

The following table describes subelements for the `security-map` element.

Table C-135 `security-map` Subelements

Element	Required	Description
`principal`	one or more	Contains the principal of the servlet or EJB client.
`user-group`	one or more	Contains the group to which the principal belongs.
`backend-principal`	only one	Specifies the user name and password required by the EIS.

Attributes

The following table describes attributes for the `security-map` element.

Table C-136 `security-map` Attributes

Attribute	Default	Description
`name`	none	Specifies a name for the security mapping.

`security-role-mapping`

Maps roles to users or groups in the currently active realm. See "Realm Configuration" in Eclipse GlassFish Application Development Guide.

The role mapping element maps a role, as specified in the EJB JAR `role-name` entries, to a environment-specific user or group. If it maps to a user, it must be a concrete user which exists in the current realm, who can log into the server using the current authentication method. If it maps to a group, the realm must support groups and the group must be a concrete group which exists in the current realm. To be useful, there must be at least one user in that realm who belongs to that group.

Superelements

`glassfish-application` (`glassfish-application.xml`), `glassfish-web-app` (`glassfish-web.xml`), `glassfish-ejb-jar` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `security-role-mapping` element.

Table C-137 `security-role-mapping` Subelements

Element	Required	Description
role-name	only one	Contains the role-name in the security-role element of the corresponding Jakarta EE deployment descriptor file.
principal-name	one or more if no group-name, otherwise zero or more	Contains a principal (user) name in the current realm. In an enterprise bean, the principal must have the run-as role specified.
group-name	one or more if no principal-name, otherwise zero or more	Contains a group name in the current realm.

service-endpoint-interface

Specifies the web service reference name relative to java:comp/env.

Superelements

port-info (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

service-impl-class

Specifies the name of the generated service implementation class.

Superelements

service-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

service-qname

Specifies the WSDL service element that is being referred to.

Superelements

service-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml); webservice-endpoint (glassfish-web.xml, glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the service-qname element.

Table C-138 service-qname subelements

Element	Required	Description
namespaceURI	only one	Specifies the namespace URI.
localpart	only one	Specifies the local part of a QNAME.

service-ref

Specifies runtime settings for a web service reference. Runtime information is only needed in the following cases:

- To define the port used to resolve a container-managed port
- To define the default Stub/Call property settings for Stub objects
- To define the URL of a final WSDL document to be used instead of the one associated with the service-ref in the standard Jakarta EE deployment descriptor

Superelements

glassfish-web-app (glassfish-web.xml), ejb (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

The following table describes subelements for the service-ref element.

Table C-139 service-ref subelements

Element	Required	Description
service-ref-name	only one	Specifies the web service reference name relative to java:comp/env.
port-info	zero or more	Specifies information for a port within a web service reference.
call-property	zero or more	Specifies JAX-RPC property values that can be set on a javax.xml.rpc.Call object before it is returned to the web service client.
wsdl-override	zero or one	Specifies a valid URL pointing to a final WSDL document.
service-impl-class	zero or one	Specifies the name of the generated service implementation class.
service-qname	zero or one	Specifies the WSDL service element that is being referenced.

service-ref-name

Specifies the web service reference name relative to `java:comp/env`.

Superelements

`service-ref` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`)

Subelements

none - contains data

servlet

Specifies a principal name for a servlet. Used for the `run-as` role defined in `web.xml`.

Superelements

`glassfish-web-app` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `servlet` element.

Table C-140 `servlet` Subelements

Element	Required	Description
`servlet-name`	only one	Contains the name of a servlet, which is matched to a `servlet-name` in `web.xml`.
`principal-name`	zero or one	Contains a principal (user) name in the current realm.
`webservice-endpoint`	zero or more	Specifies information about a web service endpoint.

servlet-impl-class

Specifies the automatically generated name of the servlet implementation class.

Superelements

`webservice-endpoint` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

servlet-name

Specifies the name of a servlet, which is matched to a servlet-name in web.xml. This name must be present in web.xml.

Superelements

cache-mapping, servlet (glassfish-web.xml)

Subelements

none - contains data

session-config

Specifies session configuration information. Overrides the web container settings for an individual web module.

Superelements

glassfish-web-app (glassfish-web.xml)

Subelements

The following table describes subelements for the session-config element.

Table C-141 session-config Subelements

Element	Required	Description
session-manager	zero or one	Specifies session manager configuration information.
session-properties	zero or one	Specifies session properties.
cookie-properties	zero or one	Specifies session cookie properties.

session-manager

Specifies session manager information.

Superelements

session-config (glassfish-web.xml)

Subelements

The following table describes subelements for the session-manager element.

Table C-142 session-manager Subelements

Element	Required	Description
manager-properties	zero or one	Specifies session manager properties.
store-properties	zero or one	Specifies session persistence (storage) properties.

Attributes

The following table describes attributes for the session-manager element.

Table C-143 session-manager Attributes

Attribute	Default	Description
persistence-type	memory	(optional) Specifies the session persistence mechanism. Allowed values are memory, file, and replicated. If you have installed and configured Coherence*Web, the coherence-web persistence type is also available. For more information, see Using Coherence*Web with Eclipse GlassFish (http://docs.oracle.com/cd/E18686_01/coh.37/e18690/glassfish.html).

session-properties

Specifies session properties.

Superelements

session-config (glassfish-web.xml)

Subelements

The following table describes subelements for the session-properties element.

Table C-144 session-properties Subelements

Element	Required	Description
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The following table describes properties for the session-properties element.

Table C-145 session-properties Properties

Property	Default	Description
timeoutSeconds	1800	Specifies the default maximum inactive interval (in seconds) for all sessions created in this web module. If set to 0 or less, sessions in this web module never expire. If a session-timeout element is specified in the web.xml file, the session-timeout value overrides any timeoutSeconds value. If neither session-timeout nor timeoutSeconds is specified, the timeoutSeconds default is used. Note that the session-timeout element in web.xml is specified in minutes, not seconds.
enableCookies	true	Uses cookies for session tracking if set to true.
enableURLRewriting	true	Enables URL rewriting. This provides session tracking via URL rewriting when the browser does not accept cookies. You must also use an encodeURL or encodeRedirectURL call in the servlet or JSP.

ssl

Defines SSL processing parameters.

Superelements

security (sun-acc.xml)

Subelements

none

Attributes

The following table describes attributes for the SSL element.

Table C-146 ssl attributes

Attribute	Default	Description
`cert-nickname`	`s1as`	(optional) The nickname of the server certificate in the certificate database or the PKCS#11 token. In the certificate, the name format is tokenname`:`nickname. Including the tokenname`:` part of the name in this attribute is optional.
`ssl2-enabled`	`false`	(optional) Determines whether SSL2 is enabled.
`ssl2-ciphers`	none	(optional) A comma-separated list of the SSL2 ciphers to be used. Ciphers not explicitly listed will be disabled for the target, even if those ciphers are available in the particular cipher suite you are using. If this option is not used, all supported ciphers are assumed to be enabled. Allowed values are `rc4`, `rc4export`, `rc2`, `rc2export`, `idea`, `des`, `desede3`.
`ssl3-enabled`	`true`	(optional) Determines whether SSL3 is enabled.
`ssl3-tls-ciphers`	none	(optional) A comma-separated list of the SSL3 and/or TLS ciphers to be used. Ciphers not explicitly listed will be disabled for the target, even if those ciphers are available in the particular cipher suite you are using. If this option is not used, all supported ciphers are assumed to be enabled. Allowed values are `SSL_RSA_WITH_RC4_128_MD5`, `SSL_RSA_WITH_3DES_EDE_CBC_SHA`, `SSL_RSA_WITH_DES_CBC_SHA`, `SSL_RSA_EXPORT_WITH_RC4_40_MD5`, `SSL_RSA_WITH_NULL_MD5`, `SSL_RSA_WITH_RC4_128_SHA`, `SSL_RSA_WITH_NULL_SHA`. Values available in previous releases are supported for backward compatibility.
`tls-enabled`	`true`	(optional) Determines whether TLS is enabled.

Attribute	Default	Description
tls-rollback-enabled	true	(optional) Determines whether TLS rollback is enabled. Enable TLS rollback for Microsoft Internet Explorer 5.0 and 5.5.

steady-pool-size

Specifies the initial and minimum number of bean instances that are maintained in the pool. Default is 32. Applies to stateless session beans and message-driven beans.

Superelements

bean-pool (glassfish-ejb-jar.xml)

Subelements

none - contains data

store-properties

Specifies session persistence (storage) properties.

Superelements

session-manager (glassfish-web.xml)

Subelements

The following table describes subelements for the store-properties element.

Table C-147 store-properties Subelements

Element	Required	Description
property (with attributes)	zero or more	Specifies a property, which has a name and a value.

Properties

The following table describes properties for the store-properties element.

Table C-148 store-properties Properties

Property	Default	Description
directory	domain-dir/generated/jsp/app-name/module-name_war	Specifies the absolute or relative pathname of the directory into which individual session files are written. A relative path is relative to the temporary work directory for this web module. Applicable only if the persistence-type attribute of the parent session-manager element is file.
persistenceScope	session	Specifies how much of the session state is stored. Allowed values are as follows: • session - The entire session state is stored every time. This mode provides the best guarantee that your session data is correctly stored for any distributable web module. • modified-session - The entire session state is stored if it has been modified. A session is considered to have been modified if HttpSession.setAttribute() or HttpSession.removeAttribute() was called. You must guarantee that setAttribute is called every time an attribute is changed. This is not a Jakarta EE specification requirement, but it is required for this mode to work properly. • modified-attribute - Only modified session attributes are stored. For this mode to work properly, you must follow some guidelines, which are explained immediately following this table. Applicable only if the persistence-type attribute of the parent session-manager element is replicated.

If the persistenceScope store property is set to modified-attribute, a web module must follow these guidelines:

- Call setAttribute every time the session state is modified.
- Make sure there are no cross-references between attributes. The object graph under each distinct attribute key is serialized and stored separately. If there are any object cross references between the objects under each separate key, they are not serialized and deserialized correctly.
- Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

stub-property

Specifies JAX-RPC property values that are set on a javax.xml.rpc.Stub object before it is returned to the web service client. The property names can be any properties supported by the JAX-RPC Stub implementation.

Superelements

port-info (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

The following table describes subelements for the stub-property element.

Table C-149 stub-property subelements

Element	Required	Description
name	only one	Specifies the name of the entity.
value	only one	Specifies the value of the entity.

Example

```
<service-ref>
<service-ref-name>service/FooProxy</service-ref-name>
  <port-info>
       <service-endpoint-interface>a.FooPort</service-endpoint-interface>
       <wsdl-port>
          <namespaceURI>urn:Foo</namespaceURI>
          <localpart>FooPort</localpart>
       </wsdl-port>
       <stub-property>
          <name>javax.xml.rpc.service.endpoint.address</name>
          <value>http://localhost:8080/a/Foo</value>
       </stub-property>
  </port-info>
</service-ref>
```

sun-cmp-mapping

Specifies beans mapped to a particular database schema.

A bean cannot be related to a bean that maps to a different database schema, even if the beans are deployed in the same EJB JAR file.

Superelements

sun-cmp-mappings (sun-cmp-mappings.xml)

Subelements

The following table describes subelements for the sun-cmp-mapping element.

Table C-150 sun-cmp-mapping Subelements

Element	Required	Description
schema	only one	Specifies the file that contains a description of the database schema.
entity-mapping	one or more	Specifies the mapping of a bean to database columns.

sun-cmp-mappings

Defines the Eclipse GlassFish specific CMP mapping configuration for an EJB JAR file. This is the root element; there can only be one sun-cmp-mappings element in a sun-cmp-mappings.xml file. See The sun-cmp-mappings.xml File.

Superelements

none

Subelements

The following table describes subelements for the sun-cmp-mappings element.

Table C-151 sun-cmp-mappings Subelements

Element	Required	Description
sun-cmp-mapping	one or more	Specifies beans mapped to a particular database schema.

table-name

Specifies the name of a database table. The table must be present in the database schema file. See "Automatic Database Schema Capture" in Eclipse GlassFish Application Development Guide.

Superelements

entity-mapping, secondary-table (sun-cmp-mappings.xml)

Subelements

none - contains data

target-server

Specifies the IIOP listener for the target server. Also specifies IIOP endpoints used for load balancing. If the Eclipse GlassFish instance on which the application client is deployed participates in a cluster, Eclipse GlassFish finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

A listener or endpoint is in the form host`:`port, where the host is an IP address or host name, and the port specifies the port number.

Not used if the deprecated `endpoints` property is defined for load balancing. For more information, see `client-container`.

Superelements

`client-container` (`sun-acc.xml`)

Subelements

The following table describes subelements for the `target-server` element.

Table C-152 `target-server` subelements

Element	Required	Description
`description`	zero or one	Specifies the description of the target server.
`security`	zero or one	Specifies the security configuration for the IIOP/SSL communication with the target server.

Attributes

The following table describes attributes for the `target-server` element.

Table C-153 `target-server` attributes

Attribute	Default	Description
`name`	none	Specifies the name of the server instance accessed by the client container.
`address`	none	Specifies the host name or IP address (resolvable by DNS) of the server to which this client attaches.
`port`	none	Specifies the naming service port number of the server to which this client attaches. For a new server instance, assign a port number other than 3700. You can change the port number in the Administration Console. Click the Help button in the Administration Console for more information.

`tie-class`

Specifies the automatically generated name of a tie implementation class for a port component.

Superelements

webservice-endpoint (glassfish-web.xml, glassfish-ejb-jar.xml)

Subelements

none - contains data

timeout

Specifies the cache-mapping specific maximum amount of time in seconds that an entry can remain in the cache after it is created or refreshed. If not specified, the default is the value of the timeout attribute of the cache element.

Superelements

cache-mapping (glassfish-web.xml)

Subelements

none - contains data

Attributes

The following table describes attributes for the timeout element.

Table C-154 timeout Attributes

Attribute	Default	Description
name	none	Specifies the timeout input parameter, whose value is interpreted in seconds. The field's type must be java.lang.Long or java.lang.Integer.
scope	request.attribute	(optional) Specifies the scope from which the input parameter is retrieved. Allowed values are context.attribute, request.header, request.parameter, request.cookie, request.attribute, and session.attribute.

transport-config

Specifies the security transport information.

Superelements

`ior-security-config` (`glassfish-ejb-jar.xml`)

Subelements

The following table describes subelements for the `transport-config` element.

Table C-155 `transport-config` Subelements

Element	Required	Description
`integrity`	only one	Specifies if the target supports integrity-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
`confidentiality`	only one	Specifies if the target supports privacy-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
`establish-trust-in-target`	only one	Specifies if the target is capable of authenticating to a client. The values are NONE, SUPPORTED, or REQUIRED.
`establish-trust-in-client`	only one	Specifies if the target is capable of authenticating a client. The values are NONE, SUPPORTED, or REQUIRED.

`transport-guarantee`

Specifies that the communication between client and server is `NONE`, `INTEGRAL`, or `CONFIDENTIAL`.

- `NONE` means the application does not require any transport guarantees.
- `INTEGRAL` means the application requires that the data sent between client and server be sent in such a way that it can't be changed in transit.
- `CONFIDENTIAL` means the application requires that the data be transmitted in a fashion that prevents other entities from observing the contents of the transmission.

In most cases, a value of `INTEGRAL` or `CONFIDENTIAL` indicates that the use of SSL is required.

Superelements

`webservice-endpoint` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

`unique-id`

Contains the unique ID for the application. This value is automatically updated each time the application is deployed or redeployed. Do not edit this value.

Superelements

`glassfish-application` (`glassfish-application.xml`), `enterprise-beans` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`url-pattern`

Specifies a servlet URL pattern for which caching is enabled. See the Servlet 2.4 specification section SRV. 11.2 for applicable patterns.

Superelements

`cache-mapping` (`glassfish-web.xml`)

Subelements

none - contains data

`user-group`

Contains the group to which the principal belongs.

Superelements

`security-map` (`glassfish-resources.xml`)

Subelements

none - contains data

`use-thread-pool-id`

Specifies the thread pool from which threads are selected for remote invocations of this bean.

Superelements

`ejb` (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`value`

Specifies the value of the entity.

Superelements

`call-property`, `stub-property` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`, `glassfish-application-client.xml`); `property` (with subelements) (`glassfish-ejb-jar.xml`)

Subelements

none - contains data

`valve`

Specifies a custom valve for this web application. You can define a valve for all the web applications on a specific virtual server. For details, see `create-virtual-server`(1).

Superelements

`glassfish-web-app` (`glassfish-web.xml`)

Subelements

The following table describes subelements for the `valve` element.

Table C-156 `valve` Subelements

Element	Required	Description
`description`	zero or one	Specifies a text description of this element.
`property` (with attributes)	zero or more	Specifies a property, which has a name and a value.

Attributes

The following table describes attributes for the `valve` element.

Table C-157 `valve` Attributes

Attribute	Default	Description
`name`	none	Specifies a unique name for the valve.
`class-name`	none	Specifies the fully qualified class name of the valve. The valve class must implement the org.apache.catalina.Valve interface from Tomcat or previous Eclipse GlassFish releases, or the org.glassfish.web.valve.GlassFishValve interface from the current Eclipse GlassFish release.

Example

```
<valve name="MyValve" classname="org.glassfish.extension.Valve">
   <property name="MyProperty1" value="MyValue1" />
   <property name="MyProperty2" value="MyValue2" />
</valve>
```

vendor

Specifies a vendor-specific icon, splash screen, text string, or a combination of these for Java Web Start download and launch screens. The complete format of this element's data is as follows:

```
<vendor>icon-image-URI::splash-screen-image-URI::vendor-text</vendor>
```

The following example vendor element contains an icon, a splash screen, and a text string:

```
<vendor>images/icon.jpg::otherDir/splash.jpg::MyCorp, Inc.</vendor>
```

The following example vendor element contains an icon and a text string:

```
<vendor>images/icon.jpg::MyCorp, Inc.</vendor>
```

The following example vendor element contains a splash screen and a text string; note the initial double colon:

```
<vendor>::otherDir/splash.jpg::MyCorp, Inc.</vendor>
```

The following example vendor element contains only a text string:

```
<vendor>MyCorp, Inc.</vendor>
```

The default value is the text string `Application Client`.

Superelements

`java-web-start-access` (`glassfish-application-client.xml`)

Subelements

none - contains data

version-identifier

Contains version information for an application or module. For more information about application versioning, see Module and Application Versions.

Superelements

glassfish-application (glassfish-application.xml), glassfish-web-app (glassfish-web-app.xml), glassfish-ejb-jar (glassfish-ejb-jar.xml), glassfish-application-client (glassfish-application-client.xml)

Subelements

none - contains data

victim-selection-policy

Specifies how stateful session beans are selected for passivation. Possible values are First In, First Out (FIFO), Least Recently Used (LRU), Not Recently Used (NRU). The default value is NRU, which is actually pseudo-LRU.

You cannot plug in your own victim selection algorithm.

The victims are generally passivated into a backup store (typically a file system or database). This store is cleaned during startup, and also by a periodic background process that removes idle entries as specified by removal-timeout-in-seconds. The backup store is monitored by a background thread (or sweeper thread) to remove unwanted entries.

Applies to stateful session beans.

Superelements

bean-cache (glassfish-ejb-jar.xml)

Subelements

none - contains data

Example

```
<victim-selection-policy>LRU</victim-selection-policy>
```

If both SSL2 and SSL3 are enabled, the server tries SSL3 encryption first. If that fails, the server tries SSL2 encryption. If both SSL2 and SSL3 are enabled for a virtual server, the server tries SSL3 encryption first. If that fails, the server tries SSL2 encryption.

web

Specifies the application's web tier configuration.

Superelements

glassfish-application (glassfish-application.xml)

Subelements

The following table describes subelements for the web element.

Table C-158 web Subelements

Element	Required	Description
web-uri	only one	Contains the web URI for the application.
context-root	only one	Contains the web context root for the web module.

web-uri

Contains the web URI for the application. Must match the corresponding element in the application.xml file.

Superelements

web (glassfish-application.xml)

Subelements

none - contains data

webservice-description

Specifies a name and optional publish location for a web service.

Superelements

glassfish-web-app (glassfish-web.xml), enterprise-beans (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the webservice-description element.

Table C-159 webservice-description subelements

Element	Required	Description
webservice-description-name	only one	Specifies a unique name for the web service within a web or EJB module.

Element	Required	Description
wsdl-publish-location	zero or one	Specifies the URL of a directory to which a web service's WSDL is published during deployment.

webservice-description-name

Specifies a unique name for the web service within a web or EJB module.

Superelements

webservice-description (glassfish-web.xml, glassfish-ejb-jar.xml)

Subelements

none - contains data

webservice-endpoint

Specifies information about a web service endpoint.

Superelements

servlet (glassfish-web.xml), ejb (glassfish-ejb-jar.xml)

Subelements

The following table describes subelements for the webservice-endpoint element.

Table C-160 webservice-endpoint subelements

Element	Required	Description
port-component-name	only one	Specifies a unique name for a port component within a web or EJB module.
endpoint-address-uri	zero or one	Specifies the automatically generated endpoint address.
login-config	zero or one	Specifies the authentication configuration for an EJB web service endpoint.
message-security-binding	zero or one	Specifies a custom authentication provider binding.
transport-guarantee	zero or one	Specifies that the communication between client and server is NONE, INTEGRAL, or CONFIDENTIAL.
service-qname	zero or one	Specifies the WSDL service element that is being referenced.
tie-class	zero or one	Specifies the automatically generated name of a tie implementation class for a port component.

Element	Required	Description
servlet-impl-class	zero or one	Specifies the automatically generated name of the generated servlet implementation class.
debugging-enabled	zero or one	Specifies whether the debugging servlet is enabled for this web service endpoint. Allowed values are true and false (the default).
property (with attributes) (glassfish-web.xml) property (with subelements) (glassfish-ejb-jar.xml)	zero or more	Specifies a property, which has a name and a value.

work-security-map

Defines a work security map, which maps a principal associated with an incoming work instance to a principal in the Eclipse GlassFish's security domain. It is possible to map multiple EIS group or user principals to the same Eclipse GlassFish principal.

This is different from a security-map, which maps the principal received during servlet or EJB authentication to the credentials accepted by the EIS.

Superelements

resources (glassfish-resources.xml)

Subelements

The following table describes subelements for the work-security-map element.

Table C-161 work-security-map Subelements

Element	Required	Description
description	zero or one	Contains a text description of this element.
principal-map	zero or more	Maps an EIS principal to a principal defined in the Eclipse GlassFish domain.
group-map	zero or more	Maps an EIS group to a group defined in the Eclipse GlassFish domain.

Attributes

The following table describes attributes for the work-security-map element.

Table C-162 work-security-map Attributes

Attribute	Default	Description
name	none	Specifies a unique name for the work security map.
description	none	Specifies a text description for this element.

wsdl-override

Specifies a valid URL pointing to a final WSDL document. If not specified, the WSDL document associated with the service-ref in the standard Jakarta EE deployment descriptor is used.

Superelements

service-ref (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

none - contains data

Example

```
// available via HTTP
<wsdl-override>http://localhost:8000/myservice/myport?WSDL</wsdl-override>

// in a file
<wsdl-override>file:/home/user1/myfinalwsdl.wsdl</wsdl-override>
```

wsdl-port

Specifies the WSDL port.

Superelements

port-info (glassfish-web.xml, glassfish-ejb-jar.xml, glassfish-application-client.xml)

Subelements

The following table describes subelements for the wsdl-port element.

Table C-163 wsdl-port subelements

Element	Required	Description
namespaceURI	only one	Specifies the namespace URI.
localpart	only one	Specifies the local part of a QNAME.

wsdl-publish-location

Specifies the URL of a directory to which a web service's WSDL is published during deployment. Any required files are published to this directory, preserving their location relative to the module-specific WSDL directory (`META-INF/wsdl` or `WEB-INF/wsdl`).

Superelements

`webservice-description` (`glassfish-web.xml`, `glassfish-ejb-jar.xml`)

Subelements

none - contains data

Example

Suppose you have an `ejb.jar` file whose `webservices.xml` file's `wsdl-file` element contains the following reference:

```
META-INF/wsdl/a/Foo.wsdl
```

Suppose your `glassfish-ejb.jar` file contains the following element:

```
<wsdl-publish-location>file:/home/user1/publish</wsdl-publish-location>
```

The final WSDL is stored in `/home/user1/publish/a/Foo.wsdl`.