Oracle® GlassFish Server 3.0.1 Application Deployment Guide

Part No: 821–1750–10
June 2010

100616@24378

Contents

Preface

This *Application Deployment Guide* describes deployment of applications and application components to the Oracle GlassFish Server, and includes information about deployment descriptors.

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

GlassFish Server 3.0.1 is developed through the GlassFish project open-source community at `https://glassfish.dev.java.net/`. The GlassFish project provides a structured process for developing the GlassFish Server platform that makes the new features of the Java EE platform available faster, while maintaining the most important feature of Java EE: compatibility. It enables Java developers to access the GlassFish Server source code and to contribute to the development of the GlassFish Server. The GlassFish project is designed to encourage communication between Oracle engineers and the community.

GlassFish Server Documentation Set

The GlassFish Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for GlassFish Server documentation is `http://docs.sun.com/coll/1343.13`. For an introduction to GlassFish Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the GlassFish Server Documentation Set

Book Title	Description
Release Notes	Provides late-breaking information about the software and the documentation. 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 GlassFish Server product.
Installation Guide	Explains how to install the software and its components.
Upgrade Guide	Explains how to upgrade to the latest version of GlassFish Server. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Administration Guide	Explains how to configure, monitor, and manage GlassFish Server 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.
Application Deployment Guide	Explains how to assemble and deploy applications to the GlassFish Server and provides information about deployment descriptors.
Your First Cup: An Introduction to the Java EE Platform	Provides a short tutorial for beginning Java EE programmers that 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.
Application Development Guide	Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the GlassFish Server. These applications follow the open Java standards model for Java EE components and APIs. This guide provides information about developer tools, security, and debugging.
Add-On Component Development Guide	Explains how to use published interfaces of GlassFish Server to develop add-on components for GlassFish Server. This document explains how to perform *only* those tasks that ensure that the add-on component is suitable for GlassFish Server.
Embedded Server Guide	Explains how to run applications in embedded GlassFish Server and to develop applications in which GlassFish Server is embedded.
Scripting Framework Guide	Explains how to develop scripting applications in languages such as Ruby on Rails and Groovy on Grails for deployment to GlassFish Server.
Troubleshooting Guide	Describes common problems that you might encounter when using GlassFish Server and how to solve them.

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

Book Title	Description
Error Message Reference	Describes error messages that you might encounter when using GlassFish Server.
Reference Manual	Provides reference information in man page format for GlassFish Server administration commands, utility commands, and related concepts.
Domain File Format Reference	Describes the format of the GlassFish Server configuration file, `domain.xml`.
Java EE 6 Tutorial	Explains how to use Java EE 6 platform technologies and APIs to develop Java EE applications.
Message Queue Release Notes	Describes new features, compatibility issues, and existing bugs for GlassFish Message Queue.
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).

Related Documentation

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

- The API specification for version 6 of Java EE is located at `http://download.oracle.com/docs/cd/E17410_01/javaee/6/api/`.
- The API specification for GlassFish Server 3.0.1, including Java EE 6 platform packages and nonplatform packages that are specific to the GlassFish Server product, is located at: `https://glassfish.dev.java.net/nonav/docs/v3/api/`.

Additionally, the following resources might be useful:

- The Java EE Specifications (`http://java.sun.com/javaee/technologies/index.jsp`)
- The Java EE Blueprints (`http://java.sun.com/reference/blueprints/`)

For information about creating enterprise applications in the NetBeans Integrated Development Environment (IDE), see `http://www.netbeans.org/kb/`.

For information about the Java DB for use with the GlassFish Server, see `http://developers.sun.com/javadb/`.

The GlassFish Samples project is a collection of sample applications that demonstrate a broad range of Java EE technologies. The GlassFish Samples are bundled with the Java EE Software Development Kit (SDK), and are also available from the GlassFish Samples project page at `https://glassfish-samples.dev.java.net/`.

Typographic Conventions

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

TABLE P–2 Typographic Conventions

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.

TABLE P–3 Symbol Conventions

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.

TABLE P–3 Symbol Conventions *(Continued)*

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

Default Paths and File Names

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

TABLE P–4 Default Paths and File Names

Placeholder	Description	Default Value
as-install	Represents the base installation directory for GlassFish Server. In configuration files, *as-install* is represented as follows: `${com.sun.aas.installRoot}`	Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: *user's-home-directory*`/glassfishv3/glassfish` Windows, all installations: *SystemDrive*`:\glassfishv3\glassfish`
as-install-parent	Represents the parent of the base installation directory for GlassFish Server.	Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: *user's-home-directory*`/glassfishv3` Windows, all installations: *SystemDrive*`:\glassfishv3`
domain-root-dir	Represents the directory in which a domain is created by default.	*as-install*`/domains/`
domain-dir	Represents the directory in which a domain's configuration is stored. In configuration files, *domain-dir* is represented as follows: `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-name*

Documentation, Support, and Training

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

- Documentation (`http://docs.sun.com/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://education.oracle.com/`)

Searching Oracle Product Documentation

Besides searching Oracle product documentation from the docs.sun.com web site, you can use a search engine by typing the following syntax in the search field:

search-term `site:docs.sun.com`

For example, to search for "broker," type the following:

```
broker site:docs.sun.com
```

To include other Oracle web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

CHAPTER 1

Overview of Oracle GlassFish Server 3.0.1 Application Deployment

Oracle GlassFish Server 3.0.1 provides an environment for developing and deploying Java applications and web services. GlassFish Server applications include Java Platform, Enterprise Edition (Java EE platform) standard features as well as features specific to GlassFish Server. This guide explains the tools and processes used for deploying applications and modules in the GlassFish Server environment. Only GlassFish Server features are described in detail in this document.

The following topics are addressed here:

- "About Application Deployment" on page 33
- "About Assembly and Deployment Events" on page 42
- "About Deployment Tools" on page 42
- "Additional Information on Application Deployment" on page 44

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 into a single unit that can be installed on an application server. The GlassFish Server assembly process conforms to the customary Java EE specifications. The only difference is that when you assemble applications or modules in GlassFish Server, you can include optional GlassFish Server deployment descriptors that enhance functionality.

Deployment is the process of using a deployment tool to specify location-specific information, such as a list of local users that can access the application, or the name of the local database. The deployment tool expands the archive file into an open directory structure that is ready for users. GlassFish Server deployment tools are described in "About Deployment Tools" on page 42.

The following topics are addressed here:

- “General Deployment Functionality” on page 34
- “Deployment Descriptors and Annotations” on page 35
- “Modules and Applications” on page 36
- “Access to Shared Framework Classes” on page 40
- “Naming Standards” on page 40

General Deployment Functionality

Various Java 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” on page 48.
- **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” on page 54.
- **Automatic Deployment.** Deploys the application archive that is placed in the autodeployment directory. For instructions, see “To Deploy an Application or Module Automatically” on page 54.
- **Directory Deployment.** Deploys the application in a directory format. For instructions, see “To Deploy an Application or Module in a Directory Format” on page 57.
- **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” on page 44 and “JSR 88 Naming” on page 41.

A deployment plan, which deploys a portable archive along with a deployment plan containing GlassFish Server deployment descriptors, can apply to any of these deployment techniques. For instructions, see “To Deploy an Application or Module by Using a Deployment Plan” on page 56.

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. For example, dynamic reloading is a quick way for developers to reload an application, but might degrade performance in a production environment. 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.

Deployment Descriptors and Annotations

A *deployment descriptor* is an XML file that describes how a Java 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. At run time, GlassFish Server reads the information in the deployment descriptor and deploys the application or module as directed.

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

- **Java EE Standard Descriptors.** Java EE standard deployment descriptors are described in the Java EE 6 specification. You can find the specification at `http://developers.sun.com/downloads/`. Information about the XML schemas that define Java EE standard deployment descriptors is available at `http://java.sun.com/xml/ns/javaee/`.
- **GlassFish Server Descriptors.** GlassFish Server provides optional deployment descriptors for configuring features that are specific to GlassFish Server. For example, when you assemble an EJB module, you annotate or create two GlassFish Server deployment descriptor files with these names: `ejb-jar.xml` and `sun-ejb-jar.xml`. For complete descriptions of these files and their elements, see Appendix B, "GlassFish Server Deployment Descriptor Files," and Appendix C, "Elements of the GlassFish Server Deployment Descriptors."

  Unless otherwise stated, settings in the GlassFish Server deployment descriptors override corresponding settings in the Java EE standard descriptors and in the GlassFish Server configuration. For more information about the domain configuration file, see *Oracle GlassFish Server 3.0.1 Domain File Format Reference*.

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. GlassFish Server supports annotation according to the following specifications:

- JSR 250 Common Annotation Specification (`http://www.jcp.org/en/jsr/detail?id=250`)
- JSR 181 Annotation for Web Services Specification (`http://www.jcp.org/en/jsr/detail?id=181`)
- EJB 3.1 Specification (`http://www.jcp.org/en/jsr/detail?id=318`)

The following annotation and deployment descriptor combinations are supported:

- Java EE applications or modules can be packaged with full Java EE 6 compliant standard and runtime deployment descriptors. If the standard deployment descriptors have specified the `metadata-complete` attribute, annotations in the module are ignored.

- Java EE applications or modules can be fully annotated with metadata defined by the listed specifications. Annotation eliminates the need for Java EE standard deployment descriptors. In most cases, the GlassFish Server deployment descriptors are also not needed.
- Java 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

A *module* is a collection of one or more Java 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" on page 36
- "Module–Based Deployment" on page 37
- "Application-Based Deployment" on page 38

Types of Modules

GlassFish Server 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 Java 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 `sun-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 `sun-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 GlassFish Server connector has annotations or a deployment descriptor file (`ra.xml`).

After deploying a J2EE connector module, you must configure it as described in Chapter 12, “Developing Connectors,” in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

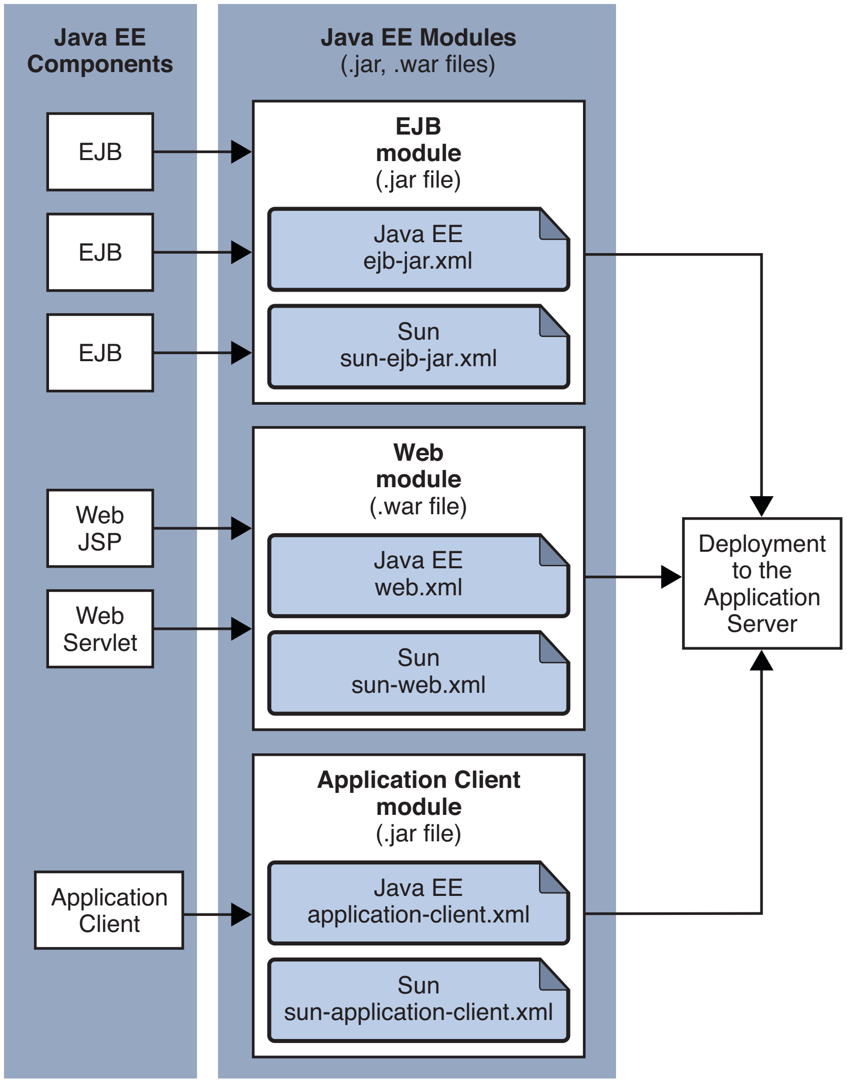
- **Application Client Module.** An application client module is a deployable software unit that consists of one or more classes, and application client deployment descriptors (`application-client.xml` and `sun-application-client.xml`). An application client JAR file applies to a GlassFish Server type of Java EE client. An application client supports the standard Java EE Application Client specifications, and, in addition, supports direct access to GlassFish Server.
- **Lifecycle Module.** A lifecycle module provides a means of running short–duration or long–duration Java-based tasks within the GlassFish Server environment. Lifecycle modules are not Java EE standard modules. See Chapter 13, “Developing Lifecycle Listeners,” in *Oracle GlassFish Server 3.0.1 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. By using package definitions in the source code of all modules, you eanble the class loader to properly locate the classes after the modules have been deployed.

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

FIGURE 1–1 Module–Based Assembly and Deployment

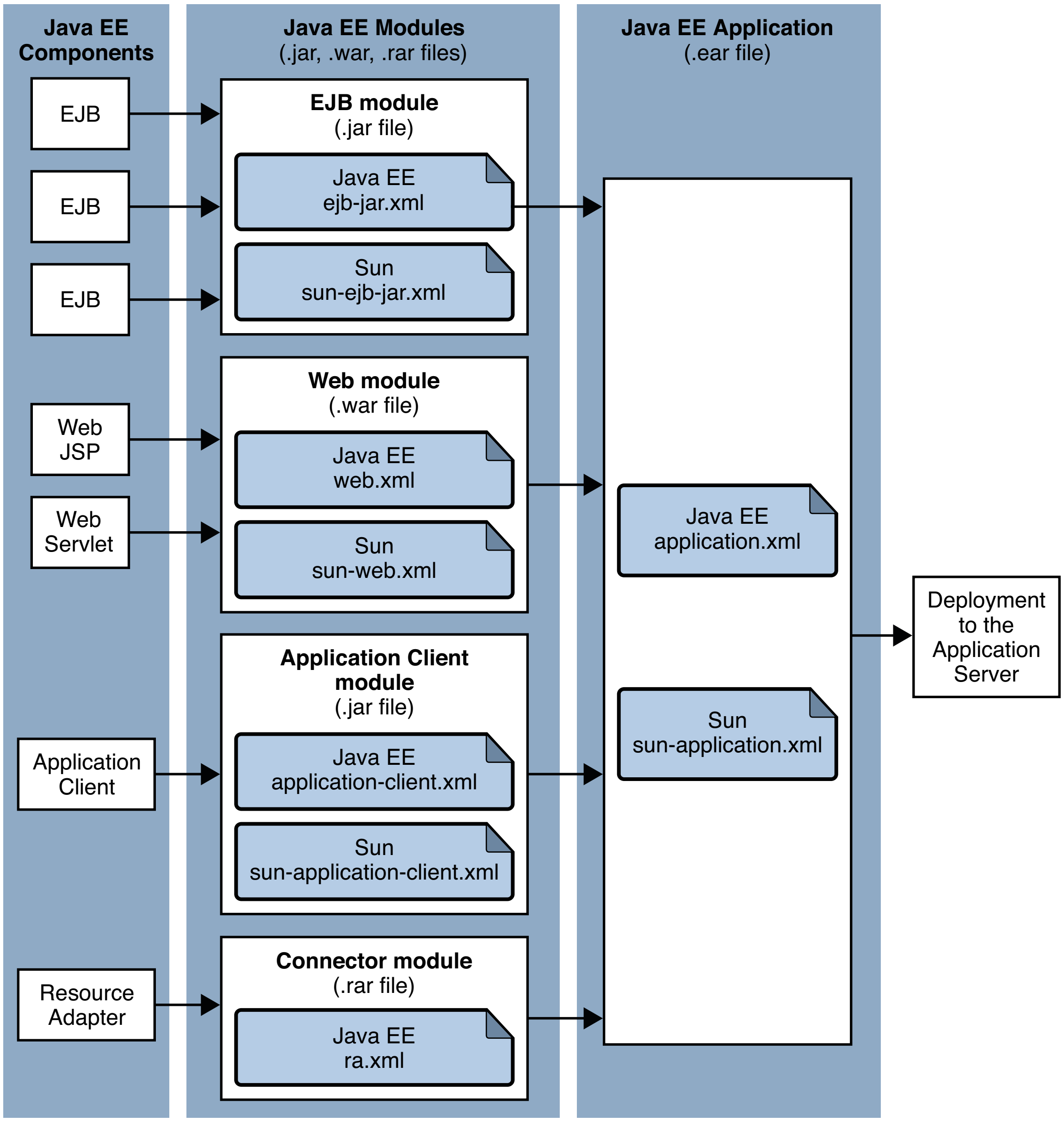


Application-Based Deployment

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 into an Enterprise archive (EAR) file which is deployed.

The following figure shows EJB, web, application client, and connector modules assembled into a Java 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 Java 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. For more information, see Chapter 2, "Class Loaders," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Note – According to the Java 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 GlassFish Server domain. Modules within an application must have unique names.

Note – Ensure that your package and file names do not contain spaces or characters that are illegal for your operating system.

You should use a Java package-like naming scheme for module 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 package-like naming scheme ensures that name collisions do not occur. The benefits of this naming practice apply not only to GlassFish Server, but to other Java EE application servers as well.

The following topics are addressed here:

- "Portable Naming" on page 40
- "JNDI Naming" on page 41
- "Directory Structure" on page 41
- "JSR 88 Naming" on page 41

Portable Naming

The Java EE 6 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>
```

This name is used before the archive name to derive the default application registration name. The application will be deployed under `xyz` if the `application.xml` file defines `application-name` as `xyz`. However, a user-specified name takes precedence over the portable application name. If `application-name` is not specified in the `application.xml` file, the application is deployed using the archive name (minus suffix) as the application registration name.

The Java EE 6 specification also defines the portable `module-name` element in the module standard deployment descriptors. For standalone modules, the defined `module-name` will be used as the application name when a name is not explicitly specified.

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 Java EE specification. During deployment, the application or module is expanded from the archive file to an open directory structure. The directories are named with `_jar` and `_war` suffixes.

If you deploy a directory, 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" on page 57.

JSR 88 Naming

There are two JSR 88 APIs that can be used to deploy applications in GlassFish Server.

If you are using the following JSR 88 API, there is no file name:

```
javax.enterprise.deploy.spi.DeploymentManager.distribute(Target[], InputStream, InputStream)
```

Because there is no file name, the name of the application is taken from the `display-name` entry in the Java EE standard deployment descriptor. If the `display-name` entry is not present, GlassFish Server creates a temporary file name and uses that name to deploy the application. Neither the Administration Console nor the `asadmin` utility uses this API.

If you are using the following preferred JSR 88 API, the name is derived from the first portion of the file name (without the `.war` or `.jar` extension):

```
javax.enterprise.deploy.spi.DeploymentManager.distribute(Target[], File, File)
```

For more information about JSR 88, see `http://jcp.org/en/jsr/detail?id=88`.

About Assembly and Deployment Events

The deployment tools that are provided by GlassFish Server can be used by anyone to deploy applications and modules into any GlassFish Server 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 Creation.** The developer creates the deployment descriptors using Java standards and tools.

   Details of the GlassFish Server deployment descriptors are contained in Appendix B, "GlassFish Server Deployment Descriptor Files," and Appendix C, "Elements of the GlassFish Server Deployment Descriptors." The GlassFish Server 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" on page 40.

   There are no GlassFish Server issues to consider.
3. **Test Deployment.** The developer performs a test deployment of the archive. For instructions, see "To Deploy an Application or Module" on page 48.
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" on page 40.
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" on page 48.
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

The following tools can be used for assembling and deploying a module or application:

- "Administration Console" on page 43
- "The `asadmin` Utility" on page 43
- " Ant Utility" on page 44
- "NetBeans IDE" on page 44

- "Eclipse IDE" on page 44
- "JSR 88 Client" on page 44

Administration Console

The GlassFish Server 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 GlassFish Server `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 Appendix A, "The `asadmin` Deployment Subcommands." All GlassFish Server `asadmin` commands are documented in *Oracle GlassFish Server 3.0.1 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 *Oracle GlassFish Server 3.0.1 Administration Guide* and the `asadmin`(1M) help page.

Ant Utility

The GlassFish Server Ant tasks can help you assemble and deploy modules and applications. For instructions, see Chapter 3, "Using Ant with GlassFish Server," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Note – GlassFish Server is compatible with Apache Ant versions 1.6.5 or greater. If you have not installed Ant, you can download it from the Update Tool. The Apache Ant Build Tool add-on component supplies Ant version 1.7.1. For information about Update Tool, see "Update Tool" in *Oracle GlassFish Server 3.0.1 Administration Guide*.

NetBeans IDE

You can use the NetBeans Integrated Development Environment (IDE), or another IDE, to assemble Java EE applications and modules. The NetBeans IDE is included in the tools bundle of the Java EE Software Development Kit (SDK). To download, see `http://java.sun.com/javaee/downloads/index.jsp`. For additional information, see `http://www.netbeans.org`.

Eclipse IDE

In addition to the bundled NetBeans IDE, a plug-in for the Eclipse IDE extends GlassFish to the Eclipse community. To download, see `https://glassfishplugins.dev.java.net/`.

JSR 88 Client

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

`deployer:Sun:AppServer::`*admin-host*`:`*admin-port*

Additional Information on Application Deployment

As specified from Java EE 6 specifications, the relevant specifications are the following:

- Java Platform, Enterprise Edition 6 Specification

  `http://jcp.org/en/jsr/detail?id=313`
- Java 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

- Java 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:

- *Oracle GlassFish Server 3.0.1 Application Development Guide*
- *Oracle GlassFish Server 3.0.1 Administration Guide*
- *Oracle GlassFish Server 3.0.1 Add-On Component Development Guide*
- *Oracle GlassFish Server 3.0.1 Reference Manual*
- GlassFish Server Administration Console online help

CHAPTER 2

Deploying Applications

This chapter provides procedures and guidelines for deploying applications and modules in the Oracle GlassFish Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

- "Deploying Applications and Modules" on page 47
- "Modifying the Configuration of a Web Application or Module" on page 58
- "Web Module Deployment Guidelines" on page 64
- "EJB Module Deployment Guidelines" on page 65
- "Deploying a Connector Module" on page 66
- "Assembling and Deploying an Application Client Module" on page 68
- "Lifecycle Module Deployment Guidelines" on page 71
- "Web Service Deployment Guidelines" on page 71

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" on page 48
- "To List Deployed Applications or Modules" on page 49
- "To Redeploy an Application or Module" on page 50
- "To Disable an Application or Module " on page 51

- "To Enable an Application or Module" on page 52
- "To Undeploy an Application or Module" on page 53
- "To Reload Changes to Applications or Modules Dynamically" on page 54
- "To Deploy an Application or Module Automatically" on page 54
- "To Deploy an Application or Module by Using a Deployment Plan" on page 56
- "To Deploy an Application or Module in a Directory Format" on page 57

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 GlassFish Server. If an error occurs during deployment, the application or module is not deployed. These failures prevent a partial deployment that could leave the server in an inconsistent state.

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. See Example 2–9. The `redeploy` subcommand also accomplishes this. You can see the status of a component by using the `show-component-status(1)` subcommand.

The Java EE 6 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>
```

This name is used before the archive name to derive the default application registration name. The application will be deployed under `xyz` if the `application.xml` file defines `application-name` as `xyz`. The user-specified name has the highest precedence. If no application-name is specified in the `application.xml` file, the application is deployed using the archive name (minus suffix) as the application registration name.

1 **Ensure that the server is running.**

Remote commands require a running server.

2 **(Optional) List deployed components by using the `list-components(1)` subcommand.**

3 **Deploy the application or module by using the `deploy(1)` subcommand.**

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

4 **(Optional) 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> 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 modules that is packaged in an RAR file.

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

▼ 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 components by using the `list-applications`(1) subcommand, the `list-components`(1) subcommand , or the `list-sub-components`(1)command.**

Information about these commands is included in these help pages.

3 **Show the status of a deployed component by using the `show-component-status`(1).**

Example 2–3 Listing Applications

The `list-applications` subcommand lists all deployed Java 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 2–4 Listing Components

The `list-components` subcommand lists the deployed Java EE components. If the `--type` option is not specified, all components are listed. This example lists the components of the deployed `cciblackbox-tx.rar`.

```
asadmin> list-components --type connector
cciblackbox-tx <connector>
Command list-components executed successfully
```

Example 2–5 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 application `MEjbApp` within module `mejb.jar`.

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

Example 2–6 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 sessions in transit at that time become invalid unless you use the `keepSessions=true` property 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(1)` subcommand or the `deploy(1)` subcommand with the `--force` option.**

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

Example 2–7 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 `keepSessions` property, active sessions of the application are retained and restored when redeployment is complete.

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

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

Example 2–8 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. The path is retrieved from the `domain.xml` file.

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

Example 2–9 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.

By default, a deployed application or module is enabled.

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(1)` 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`, `jruby`, and `webservice`.

   To see the status of deployed components, use the `show-component-status(1)` subcommand.

3. **Deactivate the application or module by using the `disable`(1) subcommand.**

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

Example 2–10 Listing Deployed Web Applications

This example lists all deployed web applications.

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

Example 2–11 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

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

1. **Ensure that the server is running.**

   Remote commands require a running server.

2 **Enable the application or module by using the `enable`(1) 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(1)` subcommand.

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

Example 2–12 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(1)` 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`, `jruby`, and `webservice`.

To see the status of deployed components, use the `show-component-status(1)` subcommand.

3 **Undeploy the application or module by using the `undeploy`(1) subcommand.**

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

Example 2–13 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 2–14 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. **Create an empty file named `.reload` at the root of the deployed application or module.**

   domain-dir`/applications/`*module-name*`/.reload`

   Note – Deployment directories might change between GlassFish Server releases.

   ---

2. **Update the timestamp of the `.reload` file to load the changes.**

   In UNIX: `touch .reload`

See Also Another method for updating an application or module is to use the `sun-appserv-update` task with the GlassFish Server Ant tasks. See Chapter 3, "Using Ant with GlassFish Server," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

▼ To Deploy an Application or Module Automatically

Note – 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 GlassFish Server 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.

1 **(Optional) Use the `set`(1) 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 **(Optional) Use the `set`(1) 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.

Note – Deployment directories might change between GlassFish Server releases.

Example 2–15 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 2–16 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 help set` at the command line.

▼ To Deploy an Application or Module by Using a Deployment Plan

In the deployment plan for an EAR file, the `sun-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).

1 **Ensure that the server is running.**

Remote commands require a running server.

2 **Deploy the application or module by using the `deploy`(1) subcommand with the `--deploymentplan` option.**

Note – Deployment directories might change between GlassFish Server releases.

Example 2–17 Deploying by 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 2–18 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 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-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 2–19 Deployment Plan Structure for a Web Module

In the deployment plan for a web module, the deployment descriptor that is specific to GlassFish Server 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 sun-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 "JSR 88 Naming" on page 41 and the JSR 88 page at `http://jcp.org/en/jsr/detail?id=88`.

▼ To Deploy an Application or Module in a Directory Format

Note – 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, 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. The directories that hold modules must be named with suffixes that identify the type of archive that the structure is based on, such as `_jar` or `_war`.

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" on page 54.

Before You Begin The directory must be accessible to the machine on which GlassFish Server runs.

On Windows, if you are deploying a directory on a mapped drive, you must be running GlassFish Server as the same user to which the mapped drive is assigned. This enables GlassFish Server 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`(1) subcommand and specifying the path to the expanded directory.**

Note – Deployment directories might change between GlassFish Server releases.

Example 2–20 Deploying an Application From a Directory

This example deploys the expanded directory for the `hello` application. The path name is stored in the domain configuration file (`domain.xml`).

```
asadmin> deploy --name hello
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" on page 59
- "To Unset a Web Context Parameter" on page 60
- "To List Web Context Parameters" on page 60
- "To Set a Web Environment Entry" on page 61
- "To Unset a Web Environment Entry" on page 62
- "To List Web Environment Entries" on page 63

▼ 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 (Java 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`(1) subcommand.**

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

Example 2–21 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 (Java 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(1)` 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`(1) subcommand.**

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

Example 2–22 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(1)` 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`(1) subcommand.**

Example 2–23 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 (Java 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`(1) subcommand.**

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

Example 2–24 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`(1) subcommand.**

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

Example 2–25 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`**(1) subcommand.**

Example 2–26 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 GlassFish Server:

- **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.
- **Data Source.** If a web application accesses a `DataSource` that is not specified in a `resource-ref` in `sun-web.xml`, or there is no `sun-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 `keepSessions` deployment property was set to true. See Example 2–7. During a server shutdown, all HTTP sessions of a web application will be persisted to the file specified by the `session-manager` property with name `sessionFilename` in the web application's `sun-web.xml` deployment descriptor. To prevent HTTP sessions from being persisted during a server shutdown, specify the `sessionFilename` property with an empty value, as follows:

```
<sun-web-app>
     <session-config>
       <session-manager>
         <manager-properties>
           <property name="sessionFilename" value="" />
         </manager-properties>
       </session-manager>
     </session-config>
   </sun-web-app>
```

  The default location where HTTP sessions will be persisted during a server shutdown is given as *domain-dir*`/generated/jsp/`*module-name*`/`*module-name*`_SESSIONS.ser`
- **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. The GlassFish Server Ant tasks `sun-appserv-deploy` and `sun-appserv-jspc` also allow you to precompile JSP files. For more information, see Chapter 3, "Using Ant with GlassFish Server," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

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

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

If you include this property when you deploy the WAR file, the generated source is kept in *domain-dir*/`generated/jsp/`*module-name*.

For more information about JSP precompilation, see "`jsp-config`" on page 162.

- **Web Context Parameters.** You can set web context parameters after deployment. See the following sections:
  - "To Set a Web Context Parameter" on page 59
  - "To Unset a Web Context Parameter" on page 60
  - "To List Web Context Parameters" on page 60
- **Web Environment Entries.** You can set web environment entries after deployment. See the following sections:
  - "To Set a Web Environment Entry" on page 61
  - "To Unset a Web Environment Entry" on page 62
  - "To List Web Environment Entries" on page 63

EJB Module Deployment Guidelines

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

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

- **JNDI Name.** If no JNDI name for the EJB JAR module is specified in the `jndi-name` element immediately under the `ejb` element in `sun-ejb-jar.xml`, or there is no `sun-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.
- **Stubs and Ties.** Use the `get-client-stubs(1)` subcommand in remote mode to retrieve stubs and ties.

Deploying a Connector Module

Deploying a standalone 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 Standalone Connector Module" on page 66
- "Redeploying a Standalone Connector Module" on page 67
- "Deploying and Configuring an Embedded Resource Adapter" on page 67

▼ To Deploy and Configure a Standalone Connector Module

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Deploy the connector module by using the `deploy`(1) 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 *Oracle GlassFish Server 3.0.1 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 *Oracle GlassFish Server 3.0.1 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 *Oracle GlassFish Server 3.0.1 Administration Guide*. If needed, you can override the default configuration properties of a resource adapter.

6. **(Optional) 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 *Oracle GlassFish Server 3.0.1 Administration Guide*.

Redeploying a Standalone 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 Chapter 2, "Class Loaders," in *Oracle GlassFish Server 3.0.1 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, GlassFish Server provides an error message.

Deploying and Configuring an Embedded Resource Adapter

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

You can create new connector connection pools and connector resources for a connector module embedded within a Java 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 GlassFish Server and the logical JNDI name used in the application component is specified in the GlassFish Server–specific XML descriptor `sun-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 GlassFish Server.

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

This section addresses the following topics:

- "To Assemble and Deploy an Application Client" on page 68
- "To Prepare Another Machine for Running an Application Client" on page 69
- "To Undeploy an Application Client" on page 70

▼ 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" on page 48.

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

   The client JAR file contains the ties and necessary classes for the application client. In this release of GlassFish Server, the client JAR file is made up of multiple files. Use either `deploy --retrieve` or `get-client-stubs`, but not both.

   - **Use the `deploy`(1) subcommand with the `--retrieve` option to retrieve the client files as part of deploying the application.**

   - **Use the `get-client-stubs`(1) subcommand to retrieve client files for a previously-deployed application.**

6 **(Optional) Test the client on the GlassFish Server 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 Java Web Start, see Chapter 11, "Developing Java Clients," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

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

▼ 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`" on page 153 element in the `sun-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" on page 40.

Before You Begin This task applies if you want 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, in UNIX: `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 **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 GlassFish Server 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 Chapter 13, "Developing Lifecycle Listeners," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

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

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.

Note – 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

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

For information about Update Tool, see "Update Tool" in *Oracle GlassFish Server 3.0.1 Administration Guide*.

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

- **Web Service Endpoint.** Deploy a web service endpoint to GlassFish Server as you would any servlet or stateless session bean. If the deployed module has a web service endpoint, the endpoint is detected automatically during deployment. The GlassFish Server -specific deployment descriptor files, `sun-web.xml` and `sun-ejb-jar.xml`, provide optional web service enhancements in their "`webservice-endpoint`" on page 247 and "`webservice-description`" on page 246 elements.
- **JSR 181 Annotated File.** Use the autodeployment feature to deploy a JSR 181 annotated file. See "To Deploy an Application or Module Automatically" on page 54.

For more information about web services, see JSR 181 (`http://www.jcp.org/en/jsr/detail?id=181`) and Chapter 6, “Developing Web Services,” in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

◆ ◆ ◆ APPENDIX A

The `asadmin` Deployment Subcommands

This appendix lists the `asadmin` deployment subcommands that are included with this release of the Oracle GlassFish Server software. For information on additional `asadmin` subcommands, see Appendix A, "Subcommands for the asadmin Utility," in *Oracle GlassFish Server 3.0.1 Administration Guide* or *Oracle GlassFish Server 3.0.1 Reference Manual*.

`deploy`(1)
: 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" on page 48.

`deploydir`(1)
: This subcommand is deprecated. Use the `deploy` subcommand instead.

`disable`(1)
: 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, see"To Disable an Application or Module " on page 51.

`enable`(1)
: 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" on page 52.

`get-client-stubs`(1)
: 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" on page 65.

`list-applications(1)`
: Lists deployed Java EE applications and modules. 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" on page 49.

`list-components(1)`
: Lists deployed Java EE components. If the `--type` option is not specified, all components are listed. Supported in remote mode only. For usage instructions, see "To List Deployed Applications or Modules" on page 49.

`list-sub-components(1)`
: 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" on page 49.

`list-web-context-param(1)`
: 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" on page 60.

`list-web-env-entry(1)`
: Lists environment entries for a deployed web application or module. Supported in remote mode only. For usage instructions, see "To List Web Environment Entries" on page 63.

`redeploy(1)`
: Overwrites an application or module that is already deployed. Supported in remote mode only. For usage instructions, see "To Redeploy an Application or Module" on page 50.

`set-web-context-param(1)`
: 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" on page 59.

`set-web-env-entry(1)`
: 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" on page 61.

`show-component-status(1)`
: 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" on page 49.

`undeploy(1)`
: Uninstalls the specified deployed application or module. Supported in remote mode only. For usage instructions, see "To Undeploy an Application or Module" on page 53.

`unset-web-context-param`(1)
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" on page 60.

`unset-web-env-entry`(1)
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" on page 62.

APPENDIX B

GlassFish Server Deployment Descriptor Files

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

The following topics are addressed here:

About the GlassFish Server 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. The DTD files for the GlassFish Server deployment descriptors are located in the *as-install*/`lib/dtds` directory.

The GlassFish Server 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`.

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

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

TABLE B–1 GlassFish Server Deployment Descriptors and DTDs

Deployment Descriptor	DTD File	Description
`sun-application.xml`	`sun-application_6_0-0.dtd`	Configures an entire Java EE application (EAR file).
`sun-web.xml`	`sun-web-app_3_0-0.dtd`	Configures a web application (WAR file).
`sun-ejb-jar.xml`	`sun-ejb-jar_3_1-0.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.
`sun-application-client.xml`	`sun-application-client_6_0-0.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. GlassFish Server provides a default file in the *domain-dir*`/config` directory. Specifying a different file is optional.
`sun-resources.xml`	`sun-resources_1_4.dtd`	Configures application-scoped resources.

The sun-application.xml File

The `sun-application.xml` file configures an entire Java EE application (EAR file). The element hierarchy is as follows:

```
sun-application
.  web
.  .  web-uri
.  .  context-root
.  pass-by-reference
.  unique-id
.  security-role-mapping
.  .  role-name
.  .  principal-name
.  .  group-name
.  realm
.  ejb-ref
.  resource-ref
.  resource-env-ref
.  service-ref
.  message-destination-ref
.  message-destination
.  archive-name
.  compatibility
```

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

```
<!DOCTYPE sun-application PUBLIC "-//Sun Microsystems, Inc.//DTD
GlassFish Application Server 3.0 Java EE Application 6.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-application_6_0-0.dtd">
<sun-application>
   <unique-id>67488732739338240</unique-id>
</sun-application>
```

The sun-web.xml File

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

```
sun-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
.  message-destination
.  .  message-destination-name
.  .  jndi-name
.  webservice-description
.  .  webservice-description-name
.  .  wsdl-publish-location
```

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

```
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
GlassFish Application Server 3.0 Servlet 3.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_3_0-0.dtd">
<sun-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/>
</sun-web-app>
```

The sun-ejb-jar.xml File

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

```
sun-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
.  .  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
```

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

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

```
<!DOCTYPE sun-ejb-jar PUBLIC "-//Sun Microsystems, Inc.//
DTD GlassFish Application Server 3.0 EJB 3.1//EN"
"http://www.sun.com/software/appserver/dtds/sun-ejb-jar_3_1-0.dtd">
<sun-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>
</sun-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 sun-application-client.xml file

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

```
sun-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
.   .   message-destination-name
.   .   jndi-name
.   message-destination-ref
.   .   message-destination-ref-name
.   .   jndi-name
.   java-web-start-access
.   .   context-root
.   .   eligible
.   .   vendor
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-application-client PUBLIC ""-//Sun Microsystems, Inc.//DTD
GlassFish Application Server 3.0 Application Client 6.0//EN""
"http://www.sun.com/software/appserver/dtds/sun-application-client_6_0-0.dtd">
<sun-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>
</sun-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. GlassFish Server 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 sun-resources.xml File

The `sun-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
```

◆ ◆ ◆ APPENDIX C

Elements of the GlassFish Server Deployment Descriptors

This appendix describes the elements of the Oracle GlassFish Server deployment 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`" on page 178 (`sun-ejb-jar.xml`)

Subelements

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

TABLE C–1 `activation-config` subelements

Element	Required	Description
"`description`" on page 135	zero or one	Specifies a text description of the activation configuration.
"`activation-config-property`" on page 96	one or more	Specifies an activation configuration property.

activation-config-property

Specifies the name and value of an activation configuration property.

Superelements

"activation-config" on page 95 (sun-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" on page 96	only one	Specifies the name of an activation configuration property.
"activation-config-property-value" on page 97	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" on page 96 (sun-ejb-jar.xml)

Subelements

none - contains data

activation-config-property-value

Specifies the value of an activation configuration property.

Superelements

"activation-config-property" on page 96 (sun-ejb-jar.xml)

Subelements

none - contains data

admin-object-resource

Defines an administered object for an inbound resource adapter.

Superelements

"resources" on page 211 (sun-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" on page 135	zero or one	Contains a text description of this element.
"property (with attributes)" on page 195	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" on page 151 (sun-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" on page 99	only one	Specifies the authentication method. The only supported value is USERNAME_PASSWORD.
"realm" on page 201	only one	Specifies the realm in which the user is authenticated.

TABLE C–5 as-context Subelements *(Continued)*

Element	Required	Description
"required" on page 205	only one	Specifies whether the authentication method specified in the auth-method element must be used for client authentication.

archive-name

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

Superelements

"sun-application" on page 229 (sun-application.xml)

Subelements

none – contains data

auth-method

Specifies the authentication method.

If the parent element is "as-context" on page 98, the only supported value is USERNAME_PASSWORD.

If the parent element is "login-config" on page 172, 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" on page 172 (sun-web.xml), "as-context" on page 98 (sun-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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"`client-container`" on page 113 (`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)" on page 195	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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

backend-principal

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

Superelements

"`security-map`" on page 217 (`sun-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`" on page 136 (`sun-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" on page 176	zero or one	Specifies the maximum number of beans allowable in cache.
"is-cache-overflow-allowed" on page 152	zero or one	Deprecated.
"cache-idle-timeout-in-seconds" on page 106	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).
"removal-timeout-in-seconds" on page 202	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" on page 206	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" on page 244	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" on page 136 (sun-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" on page 226	zero or one	Specifies the initial and minimum number of beans maintained in the pool. Default is 32.
"resize-quantity" on page 206	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.
"max-pool-size" on page 177	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" on page 177	zero or one	Deprecated.
"pool-idle-timeout-in-seconds" on page 191	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

"sun-web-app" on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the cache element.

TABLE C–11 cache Subelements

Element	Required	Description
"cache-helper" on page 105	zero or more	Specifies a custom class that implements the CacheHelper interface.
"default-helper" on page 133	zero or one	Allows you to change the properties of the default, built-in "cache-helper" on page 105 class.
"property (with attributes)" on page 195	zero or more	Specifies a cache property, which has a name and a value.
"cache-mapping" on page 107	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" on page 239 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" on page 105 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.

TABLE C–13 cache Properties *(Continued)*

Property	Default	Description
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" on page 103 (sun-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)" on page 195	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`" on page 107 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`" on page 105 used by the parent "`cache-mapping`" on page 107 element.

Superelements

"`cache-mapping`" on page 107 (`sun-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`" on page 101 (`sun-ejb-jar.xml`)

Subelements

none - contains data

cache-mapping

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

Superelements

“cache” on page 103 (sun-web.xml)

Subelements

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

TABLE C–17 cache-mapping Subelements

Element	Required	Description
“servlet-name” on page 222	requires one servlet-name or url-pattern	Contains the name of a servlet.
“url-pattern” on page 241	requires one servlet-name or url-pattern	Contains a servlet URL pattern for which caching is enabled.
“cache-helper-ref” on page 106	required if dispatcher, timeout, refresh-field, http-method, key-field, and constraint-field are not used	Contains the name of the “cache-helper” on page 105 used by the parent cache-mapping element.
“dispatcher” on page 135	zero or one if cache-helper-ref is not used	Contains a comma-separated list of RequestDispatcher methods for which caching is enabled.
“timeout” on page 239	zero or one if cache-helper-ref is not used	Contains the “cache-mapping” on page 107 specific maximum amount of time in seconds that an entry can remain in the cache after it is created or refreshed.
“refresh-field” on page 201	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” on page 150	zero or more if cache-helper-ref is not used	Contains an HTTP method that is eligible for caching.
“key-field” on page 166	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” on page 127	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`" on page 192, "`service-ref`" on page 220 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE C–18 `call-property` subelements

Element	Required	Description
"`name`" on page 186	only one	Specifies the name of the entity.
"`value`" on page 242	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`" on page 213 (`sun-ejb-jar.xml`)

Subelements

none - contains data

cert-db

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

Superelements

"security" on page 217 (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" on page 126 (sun-cmp-mappings.xml)

check-modified-at-commit

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

Superelements

"consistency" on page 126 (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" on page 126 (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" on page 119	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`” on page 136 element must be set to `true`.

Note – This element is not implemented for GlassFish Server 3.0.1.

Superelements

“`ejb`” on page 136 (`sun-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`” on page 184	one or more	Specifies a bean method.

checkpointed-methods

Deprecated. Supported for backward compatibility. Use “`checkpoint-at-end-of-method`” on page 111 instead.

Note – This element is not implemented for GlassFish Server 3.0.1.

Superelements

“`ejb`” on page 136 (`sun-ejb-jar.xml`)

class-loader

Configures the class loader for the web module.

Superelements

"sun-web-app" on page 232 (sun-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)" on page 195	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"
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 Java EE platform. **Note –** For this release, the delegate value is ignored and assumed to be set to true.

TABLE C–23 `class-loader` Attributes *(Continued)*

Attribute	Default	Description
`dynamic-reload-interval`		(optional) Not implemented. Included for backward compatibility with previous Oracle Web Server versions.

Note – 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 Java 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 GlassFish Server 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" on page 93.

Superelements

none

Subelements

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

TABLE C–25 `client-container` Subelements

Element	Required	Description
“`target-server`” on page 237	only one (developer profile)	Specifies the IIOP listener for the target server. 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`” on page 100	zero or one	Specifies the optional configuration for JAAS authentication realm.
“`client-credential`” on page 115	zero or one	Specifies the default client credential that is sent to the server.
“`log-service`” on page 171	zero or one	Specifies the default log file and the severity level of the message.
“`message-security-config`” on page 183	zero or more	Specifies configurations for message security providers.
“`property` (with attributes)” on page 195	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`” on page 237 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" on page 113 (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)" on page 195	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" on page 136 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the cmp element.

TABLE C–30 cmp Subelements

Element	Required	Description
"mapping-properties" on page 176	zero or one	This element is not implemented.
"is-one-one-cmp" on page 152	zero or one	This element is not implemented.
"one-one-finders" on page 188	zero or one	Describes the finders for CMP 1.1 beans.
"prefetch-disabled" on page 193	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" on page 143 (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" on page 147	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" on page 119	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" on page 200	zero or one	Specifies that a field is read-only.
"fetched-with" on page 145	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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"enterprise-beans" on page 142 (sun-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" on page 162	only one	Specifies the absolute jndi-name of a JDBC resource.
"default-resource-principal" on page 134	zero or one	Specifies the default runtime bindings of a resource reference.
"property (with subelements)" on page 197	zero or more	Specifies a property name and value. Used to configure PersistenceManagerFactory properties.
"create-tables-at-deploy" on page 131	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" on page 136	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" on page 132	zero or one	Specifies the name of the database vendor for which tables can be created.
"schema-generator-properties" on page 215	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" on page 143 (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" on page 118	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" on page 119	one or more	Specifies the pair of columns that determine the relationship between two database tables.
"fetched-with" on page 145	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" on page 117 (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” on page 136 (sun-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” on page 110, “cmp-field-mapping” on page 116, “column-pair” on page 119 (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” on page 117, “secondary-table” on page 216 (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" on page 119	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 GlassFish Server are B or C. Default value is B. Applies to entity beans.

Note – Commit option A is not supported for this GlassFish Server release.

Superelements

"ejb" on page 136 (sun-ejb-jar.xml)

Subelements

none - contains data

compatibility

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

Superelements

"sun-application" on page 229 (sun-application.xml)

Subelements

none – contains data

confidentiality

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

Superelements

"transport-config" on page 239 (sun-ejb-jar.xml)

Subelements

none - contains data

connector-connection-pool

Defines a connector connection pool.

Superelements

"resources" on page 211 (sun-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" on page 135	zero or one	Contains a text description of this element.
"security-map" on page 217	zero or more	Maps the principal received during servlet or EJB authentication to the credentials accepted by the EIS.
"property (with attributes)" on page 195	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: `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`" on page 125 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.
`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.
`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.

TABLE C–36 `connector-connection-pool` Attributes *(Continued)*

Attribute	Default	Description
`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.
`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.
`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.
`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.

TABLE C–36 connector-connection-pool Attributes *(Continued)*

Attribute	Default	Description
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 GlassFish Message Queue software. For a complete list of the available properties (called *administered object attributes* in the Message Queue software), see the *Oracle GlassFish Message Queue 4.4.2 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 javax.jms.TopicConnectionFactory or javax.jms.QueueConnectionFactory.
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 javax.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 javax.jms.TopicConnectionFactory or javax.jms.QueueConnectionFactory.
Password	guest	Specifies the password for connecting to the Message Queue software. For JMS resources of the Type javax.jms.TopicConnectionFactory or javax.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).

TABLE C–37 connector-connection-pool Properties *(Continued)*

Property	Default	Description
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.
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.

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

connector-resource

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

Superelements

"resources" on page 211 (sun-resources.xml)

Subelements

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

TABLE C–38 connector-resource Subelements

Element	Required	Description
"description" on page 135	zero or one	Contains a text description of this element.
"property (with attributes)" on page 195	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`" on page 121.
`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`" on page 143 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE C–40 `consistency` Subelements

Element	Required	Description
"`none`" on page 187	exactly one subelement is required	No consistency checking occurs.
"`check-modified-at-commit`" on page 110	exactly one subelement is required	Checks concurrent modification of fields in modified beans at commit time.

TABLE C–40 consistency Subelements *(Continued)*

Element	Required	Description
"lock-when-loaded" on page 170	exactly one subelement is required	Obtains an exclusive lock when the data is loaded.
"check-all-at-commit" on page 109		This element is not implemented. Do not use.
"lock-when-modified" on page 171		This element is not implemented. Do not use.
"check-version-of-accessed-instances" on page 110	exactly one subelement is required	Checks the version column of the modified beans.

constraint-field

Specifies a cacheability constraint for the given "url-pattern" on page 241 or "servlet-name" on page 222.

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" on page 107 (sun-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" on page 128	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" on page 128 subelement.
cache-on-match -failure	false	(optional) If true, caches the response if matching fails. Overrides the same attribute in a "constraint-field-value" on page 128 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" on page 127 (sun-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.

TABLE C–43 constraint-field-value Attributes *(Continued)*

Attribute	Default	Description
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" on page 153.

Superelements

"web" on page 245 (sun-application.xml), "sun-web-app" on page 232 (sun-web.xml), "java-web-start-access" on page 153 (sun-application-client.xml)

Subelements

none - contains data

cookie-properties

Specifies session cookie properties.

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

Superelements

"session-config" on page 223 (sun-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)" on page 195	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
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 GlassFish Server. You can set the path to a narrower mapping to limit the request URLs to which the browser sends cookies.
cookieMaxAgeSeconds	-1	Specifies the expiration time (in seconds) after which the browser expires the cookie.
cookieDomain	(unset)	Specifies the domain for which the cookie is valid.
cookieComment	Sun GlassFish Enterprise Server Session Tracking Cookie	Specifies the comment that identifies the session tracking cookie in the cookie file. Applications can provide a more specific comment for the cookie.
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).

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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"`cmp-resource`" on page 117 (`sun-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`" on page 211 (`sun-resources.xml`)

Subelements

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

TABLE C–46 `custom-resource` Subelements

Element	Required	Description
"`description`" on page 135	zero or one	Contains a text description of this element.
"`property` (with attributes)" on page 195	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.
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`” on page 162 subelement of the “`cmp-resource`” on page 117 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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

“`cmp-resource`” on page 117 (`sun-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” on page 247 (sun-web.xml, sun-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” on page 193 element in the sun-ejb-jar.xml file.

Superelements

“fetched-with” on page 145 (sun-cmp-mappings.xml)

Subelements

none - element is present or absent

default-helper

Passes property values to the built-in default “cache-helper” on page 105 class.

Superelements

“cache” on page 103 (sun-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)" on page 195	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`" on page 105 key generation, which concatenates the servlet path with "`key-field`" on page 166 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 *Oracle GlassFish Server 3.0.1 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 GlassFish Message Queue broker user repository. See the *Security Management* chapter in the *Oracle GlassFish Message Queue 4.4.2 Administration Guide* for details.

Superelements

"`resource-ref`" on page 209 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "`cmp-resource`" on page 117, "`mdb-connection-factory`" on page 177 (`sun-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" on page 186	only one	Specifies the default resource principal name used to sign on to a resource manager.
"password" on page 191	only one	Specifies password of the default resource principal.

description

Specifies a text description of the containing element.

Superelements

"property (with attributes)" on page 195, "valve" on page 242 (sun-web.xml); "activation-config" on page 95, "method" on page 184 (sun-ejb-jar.xml); "target-server" on page 237 (sun-acc.xml); "admin-object-resource" on page 97, "connector-connection-pool" on page 121, "connector-resource" on page 125, "custom-resource" on page 131, "external-jndi-resource" on page 144, "jdbc-connection-pool" on page 154, "jdbc-resource" on page 160, "mail-resource" on page 172, "property (with attributes)" on page 195, "resource-adapter-config" on page 207 (sun-resources.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" on page 107 (sun-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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"`cmp-resource`" on page 117 (`sun-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`" on page 142 (`sun-ejb-jar.xml`)

Subelements

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

TABLE C–51 ejb Subelements

Element	Required	Description
"ejb-name" on page 139	only one	Matches the ejb-name in the corresponding ejb-jar.xml file.
"jndi-name" on page 162	zero or more	Specifies the absolute jndi-name.
"ejb-ref" on page 139	zero or more	Maps the absolute JNDI name to the ejb-ref element in the corresponding Java EE XML file.
"resource-ref" on page 209	zero or more	Maps the absolute JNDI name to the resource-ref in the corresponding Java EE XML file.
"resource-env-ref" on page 208	zero or more	Maps the absolute JNDI name to the resource-env-ref in the corresponding Java EE XML file.
"service-ref" on page 220	zero or more	Specifies runtime settings for a web service reference.
"message-destination-ref" on page 180	zero or more	Specifies the name of a physical message destination.
"pass-by-reference" on page 189	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.
"cmp" on page 115	zero or one	Specifies runtime information for a container-managed persistence (CMP) entity bean for EJB 1.1 and EJB 2.1 beans.
"principal" on page 194	zero or one	Specifies the principal (user) name in an enterprise bean that has the run-as role specified.
"mdb-connection-factory" on page 177	zero or one	Specifies the connection factory associated with a message-driven bean.
"jms-durable-subscription-name" on page 161	zero or one	Specifies the durable subscription associated with a message-driven bean.
"jms-max-messages-load" on page 161	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" on page 151	zero or one	Specifies the security information for the IOR.
"is-read-only-bean" on page 152	zero or one	Specifies that this entity bean is read-only.
"refresh-period-in-seconds" on page 202	zero or one	Specifies the rate at which a read-only-bean must be refreshed from the data source.
"commit-option" on page 120	zero or one	Has valid values of B or C. Default value is B.
"cmt-timeout-in-seconds" on page 118	zero or one	Overrides the Transaction Timeout setting of the Transaction Service for an individual bean.
"use-thread-pool-id" on page 242	zero or one	Specifies the thread pool from which threads are selected for remote invocations of this bean.

TABLE C–51 ejb Subelements *(Continued)*

Element	Required	Description
"gen-classes" on page 148	zero or one	Specifies all the generated class names for a bean.
"bean-pool" on page 102	zero or one	Specifies the bean pool properties. Used for stateless session beans, entity beans, and message-driven beans.
"bean-cache" on page 101	zero or one	Specifies the bean cache properties. Used only for stateful session beans and entity beans.
"mdb-resource-adapter" on page 178	zero or one	Specifies runtime configuration information for a message-driven bean.
"webservice-endpoint" on page 247	zero or more	Specifies information about a web service endpoint.
"flush-at-end-of-method" on page 148	zero or one	Specifies the methods that force a database flush after execution. Used for entity beans.
"checkpointed-methods" on page 111	zero or one	Deprecated. Supported for backward compatibility. Use "checkpoint-at-end-of-method" on page 111 instead.
"checkpoint-at-end-of-method" on page 111	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`.

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 `sun-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`" on page 136, "`method`" on page 184 (`sun-ejb-jar.xml`); "`entity-mapping`" on page 143 (`sun-cmp-mappings.xml`)

Subelements

none - contains data

ejb-ref

Maps the `ejb-ref-name` in the corresponding Java 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

"sun-web-app" on page 232 (sun-web.xml), "ejb" on page 136 (sun-ejb-jar.xml), "sun-application-client" on page 230 (sun-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" on page 140	only one	Specifies the ejb-ref-name in the corresponding Java EE deployment descriptor file ejb-ref entry.
"jndi-name" on page 162	only one	Specifies the absolute jndi-name of a resource.

ejb-ref-name

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

Superelements

"ejb-ref" on page 139 (sun-web.xml, sun-ejb-jar.xml, sun-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" on page 153 (sun-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`” on page 247 (`sun-web.xml`, `sun-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

"sun-ejb-jar" on page 232 (sun-ejb-jar.xml)

Subelements

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

TABLE C–54 enterprise-beans Subelements

Element	Required	Description
"name" on page 186	zero or one	Specifies the name string.
"unique-id" on page 241	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" on page 136	zero or more	Defines runtime properties for a single enterprise bean within the application.
"pm-descriptors" on page 191	zero or one	Deprecated.
"cmp-resource" on page 117	zero or one	Specifies the database to be used for storing container-managed persistence (CMP) beans in an EJB JAR file.
"message-destination" on page 179	zero or more	Specifies the name of a logical message destination.
"webservice-description" on page 246	zero or more	Specifies a name and optional publish location for a web service.

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" on page 231 (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" on page 139	only one	Specifies the name of the entity bean in the ejb-jar.xml file to which the CMP bean corresponds.
"table-name" on page 237	only one	Specifies the name of a database table. The table must be present in the database schema file.
"cmp-field-mapping" on page 116	one or more	Associates a field with one or more columns to which it maps.
"cmr-field-mapping" on page 117	zero or more	A container-managed relationship field has a name and one or more column pairs that define the relationship.
"secondary-table" on page 216	zero or more	Describes the relationship between a bean's primary and secondary table.
"consistency" on page 126	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" on page 239 (sun-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" on page 239 (sun-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" on page 211 (sun-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`" on page 135	zero or one	Contains a text description of this element.
"`property` (with attributes)" on page 195	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 *Oracle GlassFish Server 3.0.1 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`" on page 116, 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" on page 116, 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" on page 117, 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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"cmp-field-mapping" on page 116, "cmr-field-mapping" on page 117 (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" on page 133	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" on page 166	exactly one subelement is required	Specifies the level number of a hierarchical fetch group.
"named-group" on page 187	exactly one subelement is required	Specifies the name of an independent fetch group.
"none" on page 187	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" on page 116 (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" on page 188 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the finder element.

TABLE C–59 finder Subelements

Element	Required	Description
"method-name" on page 185	only one	Specifies the method name for the finder.
"query-params" on page 200	zero or one	Specifies the query parameters for the CMP 1.1 finder.
"query-filter" on page 198	zero or one	Specifies the query filter for the CMP 1.1 finder.
"query-variables" on page 200	zero or one	Specifies variables in query expression for the CMP 1.1 finder.
"query-ordering" on page 199	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" on page 136 (sun-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" on page 184	one or more	Specifies a bean method.

gen-classes

Specifies all the generated class names for a bean.

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

Superelements

"ejb" on page 136 (sun-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" on page 203	zero or one	Specifies the fully-qualified class name of the generated EJBObject impl class.

TABLE C–61 gen-classes Subelements *(Continued)*

Element	Required	Description
"local-impl" on page 167	zero or one	Specifies the fully-qualified class name of the generated EJBLocalObject impl class.
"remote-home-impl" on page 203	zero or one	Specifies the fully-qualified class name of the generated EJBHome impl class.
"local-home-impl" on page 167	zero or one	Specifies the fully-qualified class name of the generated EJBLocalHome impl class.

group-map

Maps an EIS group to a group defined in the GlassFish Server domain.

Superelements

"work-security-map" on page 248 (sun-resources.xml)

Subelements

none

Attributes

The following table describes attributes for the group-map element.

TABLE C–62 group-map Attributes

Attribute	Default	Description
eis-group	none	Specifies an EIS group.
mapped-group	none	Specifies a group defined in the GlassFish Server domain.

group-name

Specifies a group name in the current realm.

Superelements

"security-role-mapping" on page 218 (sun-application.xml, sun-web.xml, sun-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" on page 107 (sun-web.xml)

Subelements

none - contains data

idempotent-url-pattern

Specifies a URL pattern for idempotent requests.

Superelements

"sun-web-app" on page 232 (sun-web.xml)

Subelements

none

Attributes

The following table describes attributes for the idempotent-url-pattern element.

TABLE C–63 `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`” on page 239 (`sun-ejb-jar.xml`)

Subelements

none - contains data

`ior-security-config`

Specifies the security information for the interoperable object reference (IOR).

Superelements

“`ejb`” on page 136 (`sun-ejb-jar.xml`)

Subelements

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

TABLE C–64 ior-security-config Subelements

Element	Required	Description
"transport-config" on page 239	zero or one	Specifies the security information for transport.
"as-context" on page 98	zero or one	Specifies the authentication mechanism used to authenticate the client. If specified, it is USERNAME_PASSWORD.
"sas-context" on page 213	zero or one	Describes the sas-context fields.

is-cache-overflow-allowed

This element is deprecated. Do not use.

Superelements

"bean-cache" on page 101 (sun-ejb-jar.xml)

is-one-one-cmp

This element is not used.

Superelements

"cmp" on page 115 (sun-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.

Note – This element is not implemented for GlassFish Server 3.0.1.

Superelements

"ejb" on page 136 (sun-ejb-jar.xml)

Subelements

none - contains data

java-method

Specifies a method.

Superelements

"message" on page 178 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

The following table describes subelements for the java-method element.

TABLE C–65 java-method Subelements

Element	Required	Description
"method-name" on page 185	only one	Specifies a method name.
"method-params" on page 186	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

"sun-application-client" on page 230 (sun-application-client.xml)

Subelements

The following table describes subelements for the java-web-start-access element.

TABLE C–66 java-web-start-access subelements

Element	Required	Description
"context-root" on page 129	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" on page 140	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`.
"vendor" on page 243	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`.

jdbc-connection-pool

Defines the properties that are required for creating a JDBC connection pool.

Superelements

"resources" on page 211 (`sun-resources.xml`)

Subelements

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

TABLE C–67 jdbc-connection-pool Subelements

Element	Required	Description
"description" on page 135	zero or one	Contains a text description of this element.
"property (with attributes)" on page 195	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: datasource-classname, associate-with-thread, lazy-connection-association, and lazy-connection-enlistment.

TABLE C–68 jdbc-connection-pool Attributes

Attribute	Default	Description
name	none	Specifies the name of the connection pool. A "jdbc-resource" on page 160 element's pool-name attribute refers to this name.
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. ConnectionPoolDatasource, 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. ConnectionPoolDatasource, 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.
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.
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.

TABLE C–68 jdbc-connection-pool Attributes *(Continued)*

Attribute	Default	Description
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.
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.
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. More details can be found at Connection Validation in GlassFish JDBC.
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 GlassFish Server. The specified class must implement the org.glassfish.api.jdbc.ConnectionValidation interface.

TABLE C–68 jdbc-connection-pool Attributes *(Continued)*

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.
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 Java 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 allows unlimited validation calls.
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. More details are at Connection Leak Tracing.
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.
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.

TABLE C–68 jdbc-connection-pool Attributes *(Continued)*

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. 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.
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.
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.

TABLE C–68 jdbc-connection-pool Attributes *(Continued)*

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.

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 GlassFish Server, 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 set*Name*(*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–69 jdbc-connection-pool Properties

Property	Description
user	Specifies the user name for connecting to the database.
password	Specifies the password for connecting to the database.
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.

TABLE C–69 jdbc-connection-pool Properties *(Continued)*

Property	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" on page 211 (sun-resources.xml)

Subelements

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

TABLE C–70 jdbc-resource Subelements

Element	Required	Description
"description" on page 135	zero or one	Contains a text description of this element.
"property (with attributes)" on page 195	zero or more	Specifies a property or a variable.

Attributes

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

TABLE C–71 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.
pool-name	none	Specifies the name of the associated "jdbc-connection-pool" on page 154.
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.

TABLE C–71 jdbc-resource Attributes *(Continued)*

Attribute	Default	Description
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" on page 136 (sun-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" on page 136 (sun-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" on page 95 subelement of the "mdb-resource-adapter" on page 178 element. For more information about JMS resources, see Chapter 17, "Using the Java Message Service," in *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"ejb-ref" on page 139, "message-destination" on page 179, "resource-env-ref" on page 208, "resource-ref" on page 209 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "cmp-resource" on page 117, "ejb" on page 136, "mdb-connection-factory" on page 177 (sun-ejb-jar.xml)

Subelements

none - contains data

jsp-config

Specifies JSP configuration information.

Superelements

"sun-web-app" on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the jsp-config element.

TABLE C–72 jsp-config Subelements

Element	Required	Description
"property (with attributes)" on page 195	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–73 `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 GlassFish Server'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://java.sun.com/javase/6/docs/technotes/tools/solaris/javac.html#options`.
`compilerTargetVM`	Depends on GlassFish Server'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://java.sun.com/javase/6/docs/technotes/tools/solaris/javac.html#options`.

TABLE C–73 `jsp-config` Properties *(Continued)*

Property	Default	Description
`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.
`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.
`errorOnUseBeanInvalid ClassAttribute`	`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`.
`ieClassId`	`clsid:8AD9C840-044E-11D1-B3E9-00805F499D93`	Specifies the Java plug-in COM class ID for Internet Explorer. Used by the `<jsp:plugin>` tags.
`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.

TABLE C–73 `jsp-config` Properties *(Continued)*

Property	Default	Description
`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://java.sun.com/javase/6/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.
`modification TestInterval`	`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 application	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.

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" on page 107 (sun-web.xml)

Subelements

none

Attributes

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

TABLE C–74 key-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.

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" on page 145 (sun-cmp-mappings.xml)

Subelements

none - contains data

local-home-impl

Specifies the fully-qualified class name of the generated `EJBLocalHome` `impl` class.

Note – 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`" on page 148 (`sun-ejb-jar.xml`)

Subelements

none - contains data

local-impl

Specifies the fully-qualified class name of the generated `EJBLocalObject` `impl` class.

Note – 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`" on page 148 (`sun-ejb-jar.xml`)

Subelements

none - contains data

locale-charset-info

Deprecated. For backward compatibility only. Use the “parameter-encoding” on page 188 subelement of “sun-web-app” on page 232 instead. Specifies information about the application’s internationalization settings.

Superelements

“sun-web-app” on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the locale-charset-info element.

TABLE C–75 locale-charset-info Subelements

Element	Required	Description
“locale-charset-map” on page 169	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” on page 188	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–76 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” on page 188 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" on page 188 element. For encodings, see http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html.

Superelements

"locale-charset-info" on page 168 (sun-web.xml)

Subelements

The following table describes subelements for the locale-charset-map element.

TABLE C–77 locale-charset-map Subelements

Element	Required	Description
"description" on page 135	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–78 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 GlassFish Server. 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–79 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–79 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" on page 220, "wsdl-port" on page 249 (sun-web.xml, sun-ejb-jar.xml, sun-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" on page 126 (sun-cmp-mappings.xml)

Subelements

none - element is present or absent

lock-when-modified

This element is not implemented. Do not use.

Superelements

"consistency" on page 126 (sun-cmp-mappings.xml)

log-service

Specifies configuration settings for the log file.

Superelements

"client-container" on page 113 (sun-acc.xml)

Subelements

The following table describes subelements for the log-service element.

TABLE C–80 log-service subelement

Element	Required	Description
"property (with attributes)" on page 195	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–81 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" on page 247 (sun-web.xml, sun-ejb-jar.xml)

Subelements

The following table describes subelements for the login-config element.

TABLE C–82 login-config subelements

Element	Required	Description
"auth-method" on page 99	only one	Specifies the authentication method.
"realm" on page 201	zero or one	Specifies the name of the realm used to process all authentication requests.

mail-resource

Defines a JavaMail (javax.mail.Session) resource.

Superelements

"resources" on page 211 (sun-resources.xml)

Subelements

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

TABLE C–83 mail-resource Subelements

Element	Required	Description
"description" on page 135	zero or one	Contains a text description of this element.
"property (with attributes)" on page 195	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the `mail-resource` element.

TABLE C–84 `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.
`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 JavaMail Session object later. Every property name must start with a mail- prefix. The GlassFish Server changes the dash (-) character to a period (.) in the name of the property, then saves the property to the MailConfiguration and JavaMail 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 JavaMail Session object, first edit sun-resources.xml as follows:

```
...
 <mail-resource jndi-name="mail/Session" ...>
     <property name="mail-password" value="adminadmin"/>
 </mail-resource>
 ...
```

After getting the JavaMail Session object, get the mail.password property to retrieve the value adminadmin, as follows:

```
String password = session.getProperty("mail.password");
```

For more information about JavaMail properties, see JavaMail API Documentation (http://java.sun.com/products/javamail/javadocs/index.html).

manager-properties

Specifies session manager properties.

Superelements

"session-manager" on page 223 (sun-web.xml)

Subelements

The following table describes subelements for the manager-properties element.

TABLE C–85 manager-properties Subelements

Element	Required	Description
"property (with attributes)" on page 195	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–86 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" on page 223 element is file or replicated. **Note –** The replicated persistence type is not supported in GlassFish Server 3.0.1.
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.
sessionFilename	One of the following: *domain-dir*/generated/jsp/*module-name*/*context-path*_SESSIONS.ser *domain-dir*/generated/jsp/*app-name*/*module-name*/*context-path*_SESSIONS.ser	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. To disable preservation of the session state, set this property's value to an empty string. Applicable only if the persistence-type attribute of the parent "session-manager" on page 223 element is memory. To disable this behavior and not preserve the session state, specify an empty string as the value of this property.

TABLE C–86 manager-properties Properties *(Continued)*

Property	Default	Description
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" on page 223 element is replicated. **Note –** The replicated persistence type is not supported in GlassFish Server 3.0.1.

mapping-properties

This element is not implemented.

Superelements

"cmp" on page 115 (sun-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" on page 101 (sun-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" on page 102 (sun-ejb-jar.xml)

Subelements

none - contains data

max-wait-time-in-millis

This element is deprecated. Do not use.

Superelements

"bean-pool" on page 102 (sun-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" on page 136 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the `mdb-connection-factory` element.

TABLE C–87 `mdb-connection-factory` Subelements

Element	Required	Description
"`jndi-name`" on page 162	only one	Specifies the absolute `jndi-name`.
"`default-resource-principal`" on page 134	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`" on page 136 (`sun-ejb-jar.xml`)

Subelements

The following table describes subelements for the `mdb-resource-adapter` element.

TABLE C–88 `mdb-resource-adapter` subelements

Element	Required	Description
"`resource-adapter-mid`" on page 208	zero or one	Specifies a resource adapter module ID.
"`activation-config`" on page 95	one or more	Specifies an activation configuration.

`message`

Specifies the methods or operations to which message security requirements apply.

Superelements

"`message-security`" on page 181 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

The following table describes subelements for the message element.

TABLE C–89 message Subelements

Element	Required	Description
"java-method" on page 153	zero or one	Specifies the methods or operations to which message security requirements apply.
"operation-name" on page 188	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 Java EE deployment descriptor file. Use when the message destination reference in the corresponding Java EE deployment descriptor file specifies a message-destination-link to a logical message-destination.

Superelements

"sun-web-app" on page 232 (sun-web.xml), "enterprise-beans" on page 142 (sun-ejb-jar.xml), "sun-application-client" on page 230 (sun-application-client.xml)

Subelements

The following table describes subelements for the message-destination element.

TABLE C–90 message-destination subelements

Element	Required	Description
"message-destination-name" on page 180	only one	Specifies the name of a logical message destination defined within the corresponding Java EE deployment descriptor file.
"jndi-name" on page 162	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 Java EE deployment descriptor file.

Superelements

"message-destination" on page 179 (sun-web.xml, sun-ejb-jar.xml, sun-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 Java EE deployment descriptor file does *not* specify a message-destination-link to a logical message-destination.

Superelements

"sun-web-app" on page 232 (sun-web.xml), "ejb" on page 136 (sun-ejb-jar.xml), "sun-application-client" on page 230 (sun-application-client.xml)

Subelements

The following table describes subelements for the message-destination-ref element.

TABLE C–91 message-destination-ref subelements

Element	Required	Description
"message-destination-ref-name" on page 181	only one	Specifies the name of a physical message destination defined within the corresponding Java EE deployment descriptor file.
"jndi-name" on page 162	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 Java EE deployment descriptor file.

Superelements

"message-destination-ref" on page 180 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

message-security

Specifies message security requirements.

- If the grandparent element is "webservice-endpoint" on page 247, these requirements pertain to request and response messages of the endpoint.
- If the grandparent element is "port-info" on page 192, these requirements pertain to the port of the referenced service.

Superelements

"message-security-binding" on page 182 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

The following table describes subelements for the message-security element.

TABLE C–92 message-security Subelements

Element	Required	Description
"message" on page 178	one or more	Specifies the methods or operations to which message security requirements apply.
"request-protection" on page 204	zero or one	Defines the authentication policy requirements of the application's request processing.

TABLE C–92 message-security Subelements *(Continued)*

Element	Required	Description
"response-protection" on page 212	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" on page 247 or "port-info" on page 192 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" on page 247, "port-info" on page 192 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

The following table describes subelements for the message-security-binding element.

TABLE C–93 message-security-binding Subelements

Element	Required	Description
"message-security" on page 181	zero or more	Specifies message security requirements.

Attributes

The following table describes attributes for the message-security-binding element.

TABLE C–94 message-security-binding Attributes

Attribute	Default	Description
auth-layer	none	Specifies the message layer at which authentication is performed. The value must be SOAP.

TABLE C–94 message-security-binding Attributes *(Continued)*

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" on page 113 (sun-acc.xml)

Subelements

The following table describes subelements for the message-security-config element.

TABLE C–95 message-security-config Subelements

Element	Required	Description
"provider-config" on page 197	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–96 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.
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" on page 111, "flush-at-end-of-method" on page 148 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the method element.

TABLE C–97 method Subelements

Element	Required	Description
"description" on page 135	zero or one	Specifies an optional text description.
"ejb-name" on page 139	zero or one	Matches the ejb-name in the corresponding ejb-jar.xml file.
"method-name" on page 185	only one	Specifies a method name.
"method-intf" on page 184	zero or one	Specifies the method interface to distinguish between methods with the same name in different interfaces.
"method-params" on page 186	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" on page 184 (sun-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" on page 153 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "finder" on page 147, "query-method" on page 199 , "method" on page 184 (sun-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" on page 186 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

method-params

Specifies fully qualified Java type names of method parameters.

Superelements

"java-method" on page 153 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "query-method" on page 199, "method" on page 184 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the method-params element.

TABLE C–98 method-params Subelements

Element	Required	Description
"method-param" on page 185	zero or more	Specifies the fully qualified Java type name of a method parameter.

name

Specifies the name of the entity.

Superelements

"call-property" on page 108, "default-resource-principal" on page 134, "stub-property" on page 228 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "enterprise-beans" on page 142, "principal" on page 194, "property (with subelements)" on page 197 (sun-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" on page 145 (sun-cmp-mappings.xml)

Subelements

none - contains data

namespaceURI

Specifies the namespace URI.

Superelements

"service-qname" on page 220, "wsdl-port" on page 249 (sun-web.xml, sun-ejb-jar.xml, sun-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" on page 126, "fetched-with" on page 145 (sun-cmp-mappings.xml)

Subelements

none - element is present or absent

one-one-finders

Describes the finders for CMP 1.1 beans.

Superelements

"cmp" on page 115 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the one-one-finders element.

TABLE C–99 one-one-finders Subelements

Element	Required	Description
"finder" on page 147	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" on page 178 (sun-web.xml, sun-ejb-jar.xml, sun-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 "sun-web-app" on page 232 and "locale-charset-info" on page 168 elements have parameter-encoding subelements, the subelement of sun-web-app takes precedence. For encodings, see http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html.

Superelements

"locale-charset-info" on page 168, "sun-web-app" on page 232 (sun-web.xml)

Subelements

none

Attributes

The following table describes attributes for the `parameter-encoding` element.

TABLE C–100 `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.

Note – 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 GlassFish Server 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 `sun-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 `sun-ejb-jar.xml` file. If `pass-by-reference` is used at both the bean and application level, the bean level takes precedence.

Superelements

"`sun-application`" on page 229 (`sun-application.xml`), "`ejb`" on page 136 (`sun-ejb-jar.xml`)

Subelements

none - contains data

password

Specifies the password for the principal.

Superelements

"default-resource-principal" on page 134 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

pm-descriptors

This element and its subelements are deprecated. Do not use.

Superelements

"enterprise-beans" on page 142 (sun-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.

Note – For a stateless session bean or a message-driven bean, the bean is removed (garbage collected) when the timeout expires.

Superelements

"bean-pool" on page 102 (sun-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" on page 247 (sun-web.xml, sun-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" on page 220 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

The following table describes subelements for the port-info element.

TABLE C–101 port-info subelements

Element	Required	Description
"service-endpoint-interface" on page 219	zero or one	Specifies the web service reference name relative to java:comp/env.
"wsdl-port" on page 249	zero or one	Specifies the WSDL port.
"stub-property" on page 228	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" on page 108	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" on page 182	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" on page 145 element is set to "default" on page 133.

Superelements

"cmp" on page 115 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the prefetch-disabled element.

TABLE C–102 prefetch-disabled Subelements

Element	Required	Description
"query-method" on page 199	one or more	Specifies a query method.

`principal`

Defines a user name on the platform.

Superelements

"`ejb`" on page 136 (`sun-ejb-jar.xml`); "`security-map`" on page 217 (`sun-resources.xml`)

Subelements

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

TABLE C–103 `principal` Subelements

Element	Required	Description
"`name`" on page 186	only one	Specifies the name of the user.

`principal-map`

Maps an EIS principal to a principal defined in the GlassFish Server domain.

Superelements

"`work-security-map`" on page 248 (`sun-resources.xml`)

Subelements

none

Attributes

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

TABLE C–104 `principal-map` Attributes

Attribute	Default	Description
`eis-principal`	none	Specifies an EIS principal.

TABLE C–104 principal-map Attributes *(Continued)*

Attribute	Default	Description
mapped-principal	none	Specifies a principal defined in the GlassFish Server 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" on page 218 (sun-application.xml, sun-web.xml, sun-ejb-jar.xml), "servlet" on page 221 (sun-web.xml)

Subelements

none - contains data

Attributes

The following table describes attributes for the principal-name element.

TABLE C–105 principal-name Attributes

Attribute	Default	Description
class-name	com.sun.enterprise.deployment.PrincipalImpl	(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 GlassFish Server
- Needed by a system or object that GlassFish Server doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

"cache" on page 103, "cache-helper" on page 105, "class-loader" on page 112, "cookie-properties" on page 129, "default-helper" on page 133, "manager-properties" on page 174, "session-properties" on page 224, "store-properties" on page 226, "sun-web-app" on page 232, "valve" on page 242, "webservice-endpoint" on page 247 (sun-web.xml); "auth-realm" on page 100, "client-container" on page 113, "client-credential" on page 115, "log-service" on page 171, "provider-config" on page 197 (sun-acc.xml); "admin-object-resource" on page 97, "connector-connection-pool" on page 121, "connector-resource" on page 125, "custom-resource" on page 131, "external-jndi-resource" on page 144, "jdbc-connection-pool" on page 154, "jdbc-resource" on page 160, "mail-resource" on page 172, "resource-adapter-config" on page 207 (sun-resources.xml)

Subelements

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

TABLE C–106 `property` Subelements

Element	Required	Description
"description" on page 135	zero or one	Specifies an optional text description of a property.

Note – The `property` element in the `sun-acc.xml` file has no subelements.

Attributes

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

TABLE C–107 `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 GlassFish Server
- Needed by a system or object that GlassFish Server doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

"cmp-resource" on page 117, "schema-generator-properties" on page 215, "webservice-endpoint" on page 247 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the property element.

TABLE C–108 property subelements

Element	Required	Description
"name" on page 186	only one	Specifies the name of the property.
"value" on page 242	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" on page 183 (sun-acc.xml)

Subelements

The following table describes subelements for the provider-config element.

TABLE C–109 provider-config Subelements

Element	Required	Description
"request-policy" on page 204	zero or one	Defines the authentication policy requirements of the authentication provider's request processing.
"response-policy" on page 212	zero or one	Defines the authentication policy requirements of the authentication provider's response processing.
"property (with attributes)" on page 195	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–110 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.
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" on page 147 (sun-ejb-jar.xml)

Subelements

none - contains data

query-method

Specifies a query method.

Superelements

"prefetch-disabled" on page 193 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the query-method element.

TABLE C–111 query-method Subelements

Element	Required	Description
"method-name" on page 185	only one	Specifies a method name.
"method-params" on page 186	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" on page 147 (sun-ejb-jar.xml)

Subelements

none - contains data

query-params

Specifies the query parameters for the CMP 1.1 finder.

Superelements

"finder" on page 147 (sun-ejb-jar.xml)

Subelements

none - contains data

query-variables

Specifies variables in the query expression for the CMP 1.1 finder.

Superelements

"finder" on page 147 (sun-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" on page 116 (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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"`sun-application`" on page 229 (`sun-application.xml`), "`as-context`" on page 98, "`login-config`" on page 172 (`sun-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`" on page 107 (`sun-web.xml`)

Subelements

none

Attributes

The following table describes attributes for the `refresh-field` element.

TABLE C–112 `refresh-field` Attributes

Attribute	Default	Description
`name`	none	Specifies the input parameter name.

TABLE C–112 refresh-field Attributes *(Continued)*

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.

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" on page 136 (sun-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" on page 106.

Superelements

"bean-cache" on page 101 (sun-ejb-jar.xml)

Subelements

none - contains data

remote-home-impl

Specifies the fully-qualified class name of the generated EJBHome impl class.

Note – 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" on page 148 (sun-ejb-jar.xml)

Subelements

none - contains data

remote-impl

Specifies the fully-qualified class name of the generated EJBObject impl class.

Note – 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" on page 148 (sun-ejb-jar.xml)

Subelements

none - contains data

request-policy

Defines the authentication policy requirements of the authentication provider's request processing.

Superelements

"provider-config" on page 197 (sun-acc.xml)

Subelements

none

Attributes

The following table describes attributes for the request-policy element.

TABLE C–113 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" on page 181 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none

Attributes

The following table describes attributes for the request-protection element.

TABLE C–114 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" on page 99 element must be used for client authentication. The value is true or false (the default).

Superelements

"as-context" on page 98 (sun-ejb-jar.xml)

Subelements

none - contains data

res-ref-name

Specifies the res-ref-name in the corresponding Java 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" on page 209 (sun-web.xml, sun-ejb-jar.xml, sun-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" on page 226 quantity of beans (applies to pools only for creation). If the pool has more than steady-pool-size minus "resize-quantity" on page 206 of beans, then resize-quantity is still created.
- Removed, when the "pool-idle-timeout-in-seconds" on page 191 timer expires and a cleaner thread removes any unused instances.
  - For caches, when "max-cache-size" on page 176 is reached, resize-quantity beans are selected for passivation using the "victim-selection-policy" on page 244. In addition, the "cache-idle-timeout-in-seconds" on page 106 or "removal-timeout-in-seconds" on page 202 timers passivate beans from the cache.
  - For pools, when the "max-pool-size" on page 177 is reached, resize-quantity beans are selected for removal. In addition, the "pool-idle-timeout-in-seconds" on page 191 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" on page 101, "bean-pool" on page 102 (sun-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`" on page 211 (`sun-resources.xml`)

Subelements

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

TABLE C–115 `resource-adapter-config` Subelements

Element	Required	Description
"`property` (with attributes)" on page 195	zero or more	Specifies a property or a variable.

Attributes

The following table describes attributes for the `resource-adapter-config` element.

TABLE C–116 `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`" on page 178 (`sun-ejb-jar.xml`)

Subelements

none - contains data

resource-env-ref

Maps the `res-ref-name` in the corresponding Java EE deployment descriptor file `resource-env-ref` entry to the absolute `jndi-name` of a resource.

Superelements

"`sun-web-app`" on page 232 (`sun-web.xml`), "`ejb`" on page 136 (`sun-ejb-jar.xml`), "`sun-application-client`" on page 230 (`sun-application-client.xml`)

Subelements

The following table describes subelements for the `resource-env-ref` element.

TABLE C–117 resource-env-ref Subelements

Element	Required	Description
“resource-env-ref-name” on page 209	only one	Specifies the res-ref-name in the corresponding Java EE deployment descriptor file resource-env-ref entry.
“jndi-name” on page 162	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 Java EE deployment descriptor file resource-env-ref entry.

Superelements

“resource-env-ref” on page 208 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

resource-ref

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

Note – Connections acquired from JMS connection factories are not shareable in the current release of GlassFish Server. 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 GlassFish Message Queue, the `default-resource-principal` (name/password) must exist in the Message Queue user repository. Refer to the *Security Management* chapter in the *Oracle GlassFish Message Queue 4.4.2 Administration Guide* for information on how to manage the Message Queue user repository.

Superelements

"`sun-web-app`" on page 232 (`sun-web.xml`), "`ejb`" on page 136 (`sun-ejb-jar.xml`), "`sun-application-client`" on page 230 (`sun-application-client.xml`)

Subelements

The following table describes subelements for the `resource-ref` element.

TABLE C–118 `resource-ref` Subelements

Element	Required	Description
"`res-ref-name`" on page 205	only one	Specifies the `res-ref-name` in the corresponding Java EE deployment descriptor file `resource-ref` entry.
"`jndi-name`" on page 162	only one	Specifies the absolute `jndi-name` of a resource.
"`default-resource-principal`" on page 134	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 `sun-resources.xml` file. See "The sun-resources.xml File" on page 93.

Note – 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 GlassFish Server application should begin with the string `java:app/`.

Superelements

none

Subelements

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

TABLE C–119 `resources` Subelements

Element	Required	Description
"`custom-resource`" on page 131	zero or more	Defines a custom resource.
"`external-jndi-resource`" on page 144	zero or more	Defines a resource that resides in an external JNDI repository.
"`jdbc-resource`" on page 160	zero or more	Defines a JDBC (Java Database Connectivity) resource.
"`mail-resource`" on page 172	zero or more	Defines a JavaMail resource.
"`admin-object-resource`" on page 97	zero or more	Defines an administered object for an inbound resource adapter.
"`connector-resource`" on page 125	zero or more	Defines a connector (resource adapter) resource.
"`resource-adapter-config`" on page 207	zero or more	Defines a resource adapter configuration.
"`jdbc-connection-pool`" on page 154	zero or more	Defines the properties that are required for creating a JDBC connection pool.
"`connector-connection-pool`" on page 121	zero or more	Defines the properties that are required for creating a connector connection pool.
"`work-security-map`" on page 248	zero or more	Defines a work security map.

Note – 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`" on page 197 (`sun-acc.xml`)

Subelements

none

Attributes

The following table describes attributes for the `response-policy` element.

TABLE C–120 `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`" on page 181 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

none

Attributes

The following table describes attributes for the response-protection element.

TABLE C–121 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).
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 Java EE deployment descriptor file.

Superelements

"security-role-mapping" on page 218 (sun-application.xml, sun-web.xml, sun-ejb-jar.xml)

Subelements

none - contains data

sas-context

Describes the sas-context fields.

Superelements

"ior-security-config" on page 151 (sun-ejb-jar.xml)

Subelements

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

TABLE C–122 `sas-context` Subelements

Element	Required	Description
"`caller-propagation`" on page 108	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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"`sun-cmp-mapping`" on page 231 (`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" on page 117 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the schema-generator-properties element.

TABLE C–123 schema-generator-properties Subelements

Element	Required	Description
"property (with subelements)" on page 197	zero or more	Specifies a property name and value.

Properties

The following table describes properties for the schema-generator-properties element.

TABLE C–124 schema-generator-properties Properties

Property	Default	Description
use-unique-table-names	false	Specifies that generated table names are unique within each GlassFish Server domain. This property can be overridden during deployment. See "Generation Options for CMP" in *Oracle GlassFish Server 3.0.1 Application Development Guide*.
bean-name.*field-name*.*attribute*	none	Defines a column attribute. For attribute descriptions, see Table C–125.

The following table lists the column attributes for properties defined in the schema-generator-properties element.

TABLE C–125 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.

TABLE C–125 `schema-generator-properties` Column Attributes *(Continued)*

Attribute	Description
`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`" on page 143 (`sun-cmp-mappings.xml`)

Subelements

The following table describes subelements for the `secondary-table` element.

TABLE C–126 secondary table Subelements

Element	Required	Description
"table-name" on page 237	only one	Specifies the name of a database table.
"column-pair" on page 119	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" on page 237 (sun-acc.xml)

Subelements

The following table describes subelements for the security element.

TABLE C–127 security Subelements

Element	Required	Description
"ssl" on page 225	only one	Specifies the SSL processing parameters.
"cert-db" on page 109	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 GlassFish Server principals to the same back-end principal.

This is different from a "work-security-map" on page 248, which maps a principal associated with an incoming work instance to a principal in the GlassFish Server's security domain.

Superelements

"connector-connection-pool" on page 121 (sun-resources.xml)

Subelements

The following table describes subelements for the `security-map` element.

TABLE C–128 `security-map` Subelements

Element	Required	Description
"`principal`" on page 194	one or more	Contains the principal of the servlet or EJB client.
"`user-group`" on page 241	one or more	Contains the group to which the principal belongs.
"`backend-principal`" on page 101	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–129 `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 *Oracle GlassFish Server 3.0.1 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

"`sun-application`" on page 229 (`sun-application.xml`), "`sun-web-app`" on page 232 (`sun-web.xml`), "`sun-ejb-jar`" on page 232 (`sun-ejb-jar.xml`)

Subelements

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

TABLE C–130 security-role-mapping Subelements

Element	Required	Description
"role-name" on page 213	only one	Contains the role-name in the security-role element of the corresponding Java EE deployment descriptor file.
"principal-name" on page 195	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" on page 150	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" on page 192 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

service-impl-class

Specifies the name of the generated service implementation class.

Superelements

"service-ref" on page 220 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

none - contains data

service-qname

Specifies the WSDL service element that is being referred to.

Superelements

"service-ref" on page 220 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "webservice-endpoint" on page 247 (sun-web.xml, sun-ejb-jar.xml)

Subelements

The following table describes subelements for the service-qname element.

TABLE C–131 service-qname subelements

Element	Required	Description
"namespaceURI" on page 187	only one	Specifies the namespace URI.
"localpart" on page 170	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 Java EE deployment descriptor

Superelements

"sun-web-app" on page 232 (sun-web.xml), "ejb" on page 136 (sun-ejb-jar.xml), "sun-application-client" on page 230 (sun-application-client.xml)

Subelements

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

TABLE C–132 service-ref subelements

Element	Required	Description
“service-ref-name” on page 221	only one	Specifies the web service reference name relative to java:comp/env.
“port-info” on page 192	zero or more	Specifies information for a port within a web service reference.
“call-property” on page 108	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” on page 249	zero or one	Specifies a valid URL pointing to a final WSDL document.
“service-impl-class” on page 219	zero or one	Specifies the name of the generated service implementation class.
“service-qname” on page 220	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” on page 220 (sun-web.xml, sun-ejb-jar.xml, sun-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

“sun-web-app” on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the servlet element.

TABLE C–133 servlet Subelements

Element	Required	Description
"servlet-name" on page 222	only one	Contains the name of a servlet, which is matched to a servlet-name in web.xml.
"principal-name" on page 195	zero or one	Contains a principal (user) name in the current realm.
"webservice-endpoint" on page 247	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" on page 247 (sun-web.xml, sun-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" on page 107, "servlet" on page 221 (sun-web.xml)

Subelements

none - contains data

session-config

Specifies session configuration information. Overrides the web container settings for an individual web module.

Superelements

"sun-web-app" on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the session-config element.

TABLE C–134 session-config Subelements

Element	Required	Description
"session-manager" on page 223	zero or one	Specifies session manager configuration information.
"session-properties" on page 224	zero or one	Specifies session properties.
"cookie-properties" on page 129	zero or one	Specifies session cookie properties.

session-manager

Specifies session manager information.

Superelements

"session-config" on page 223 (sun-web.xml)

Subelements

The following table describes subelements for the session-manager element.

TABLE C–135 session-manager Subelements

Element	Required	Description
"manager-properties" on page 174	zero or one	Specifies session manager properties.
"store-properties" on page 226	zero or one	Specifies session persistence (storage) properties.

Attributes

The following table describes attributes for the `session-manager` element.

TABLE C–136 `session-manager` Attributes

Attribute	Default	Description
`persistence-type`	`memory`	(optional) Specifies the session persistence mechanism. Allowed values are `memory`, `file`, and `replicated`. **Note –** The `replicated` persistence type is not supported in GlassFish Server 3.0.1.

`session-properties`

Specifies session properties.

Superelements

"`session-config`" on page 223 (`sun-web.xml`)

Subelements

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

TABLE C–137 `session-properties` Subelements

Element	Required	Description
"`property` (with attributes)" on page 195	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–138 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" on page 217 (sun-acc.xml)

Subelements

none

Attributes

The following table describes attributes for the SSL element.

TABLE C–139 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.

TABLE C–139 ssl attributes *(Continued)*

Attribute	Default	Description
ssl2-ciphers	none	(optional) A space-separated list of the SSL2 ciphers used with the prefix + to enable or - to disable. For example, +rc4. 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 space-separated list of the SSL3 ciphers used, with the prefix + to enable or - to disable, for example +SSL_RSA_WITH_RC4_128_MD5. 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, and SSL_RSA_WITH_NULL_SHA. Values available in previous releases are supported for backward compatibility.
tls-enabled	true	(optional) Determines whether TLS is enabled.
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" on page 102 (sun-ejb-jar.xml)

Subelements

none - contains data

store-properties

Specifies session persistence (storage) properties.

Superelements

"session-manager" on page 223 (sun-web.xml)

Subelements

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

TABLE C–140 `store-properties` Subelements

Element	Required	Description
"`property` (with attributes)" on page 195	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–141 `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`" on page 223 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 Java 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`" on page 223 element is `replicated`. **Note –** The `replicated` persistence type is not supported in GlassFish Server 3.0.1.

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" on page 192 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

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

TABLE C–142 stub-property subelements

Element	Required	Description
"name" on page 186	only one	Specifies the name of the entity.
"value" on page 242	only one	Specifies the value of the entity.

Properties

The following table describes properties for the stub-property element.

TABLE C–143 stub-property properties

Property	Default	Description
jbi-enabled	true	Determines whether the visibility of this endpoint as a Java Business Integration service is enabled or disabled.

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-application

Defines the GlassFish Server specific configuration for an application. This is the root element; there can only be one `sun-application` element in a `sun-application.xml` file. See "The sun-application.xml File" on page 78.

Superelements

none

Subelements

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

TABLE C–144 `sun-application` Subelements

Element	Required	Description
"`web`" on page 245	zero or more	Specifies the application's web tier configuration.
"`pass-by-reference`" on page 189	zero or one	Determines whether EJB modules use pass-by-value or pass-by-reference semantics.
"`unique-id`" on page 241	zero or one	Contains the unique ID for the application.
"`security-role-mapping`" on page 218	zero or more	Maps a role in the corresponding Java EE XML file to a user or group.

TABLE C–144 sun-application Subelements *(Continued)*

Element	Required	Description
"realm" on page 201	zero or one	Specifies an authentication realm.

sun-application-client

Defines the GlassFish Server specific configuration for an application client. This is the root element; there can only be one sun-application-client element in a sun-application-client.xml file. See "The sun-application-client.xml file" on page 91.

Superelements

none

Subelements

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

TABLE C–145 sun-application-client subelements

Element	Required	Description
"ejb-ref" on page 139	zero or more	Maps the absolute JNDI name to the ejb-ref in the corresponding Java EE XML file.
"resource-ref" on page 209	zero or more	Maps the absolute JNDI name to the resource-ref in the corresponding Java EE XML file.
"resource-env-ref" on page 208	zero or more	Maps the absolute JNDI name to the resource-env-ref in the corresponding Java EE XML file.
"service-ref" on page 220	zero or more	Specifies runtime settings for a web service reference.
"message-destination" on page 179	zero or more	Specifies the name of a logical message destination.
"message-destination-ref" on page 180	zero or more	Specifies the name of a physical message destination.
"java-web-start-access" on page 153	zero or one	Specifies changes to default Java Web Start parameters.

sun-cmp-mapping

Specifies beans mapped to a particular database schema.

Note – 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" on page 231 (sun-cmp-mappings.xml)

Subelements

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

TABLE C–146 sun-cmp-mapping Subelements

Element	Required	Description
"schema" on page 214	only one	Specifies the file that contains a description of the database schema.
"entity-mapping" on page 143	one or more	Specifies the mapping of a bean to database columns.

sun-cmp-mappings

Defines the GlassFish Server 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" on page 87.

Superelements

none

Subelements

The following table describes subelements for the sun-cmp-mappings element.

TABLE C–147 sun-cmp-mappings Subelements

Element	Required	Description
"sun-cmp-mapping" on page 231	one or more	Specifies beans mapped to a particular database schema.

sun-ejb-jar

Defines the GlassFish Server specific configuration for an EJB JAR file. This is the root element; there can only be one `sun-ejb-jar` element in a `sun-ejb-jar.xml` file. See "The sun-ejb-jar.xml File" on page 82.

Superelements

none

Subelements

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

TABLE C–148 sun-ejb-jar Subelements

Element	Required	Description
"security-role-mapping" on page 218	zero or more	Maps a role in the corresponding Java EE XML file to a user or group.
"enterprise-beans" on page 142	only one	Describes all the runtime properties for an EJB JAR file in the application.

sun-web-app

Defines GlassFish Server specific configuration for a web module. This is the root element; there can only be one `sun-web-app` element in a `sun-web.xml` file. See "The sun-web.xml File" on page 79.

Superelements

none

Subelements

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

TABLE C–149 sun-web-app Subelements

Element	Required	Description
"context-root" on page 129	zero or one	Contains the web context root for the web module.
"security-role-mapping" on page 218	zero or more	Maps roles to users or groups in the currently active realm.
"servlet" on page 221	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" on page 150	zero or more	Specifies a URL pattern for idempotent requests.
"session-config" on page 223	zero or one	Specifies session manager, session cookie, and other session-related information.
"ejb-ref" on page 139	zero or more	Maps the absolute JNDI name to the ejb-ref in the corresponding Java EE XML file.
"resource-ref" on page 209	zero or more	Maps the absolute JNDI name to the resource-ref in the corresponding Java EE XML file.
"resource-env-ref" on page 208	zero or more	Maps the absolute JNDI name to the resource-env-ref in the corresponding Java EE XML file.
"service-ref" on page 220	zero or more	Specifies runtime settings for a web service reference.
"message-destination-ref" on page 180	zero or more	Specifies the name of a physical message destination.
"cache" on page 103	zero or one	Configures caching for web application components.
"class-loader" on page 112	zero or one	Specifies class loader configuration information.
"jsp-config" on page 162	zero or one	Specifies JSP configuration information.
"locale-charset-info" on page 168	zero or one	Deprecated. Use the parameter-encoding subelement of sun-web-app instead.
"parameter-encoding" on page 188	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)" on page 195	zero or more	Specifies a property, which has a name and a value.
"valve" on page 242	zero or more	Specifies a custom valve.
"message-destination" on page 179	zero or more	Specifies the name of a logical message destination.
"webservice-description" on page 246	zero or more	Specifies a name and optional publish location for a web service.

Attributes

The following table describes attributes for the `sun-web-app` element.

TABLE C–150 `sun-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 `sun-web-app` element.

TABLE C–151 `sun-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.

TABLE C–151 `sun-web-app` Properties *(Continued)*

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 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)`.

TABLE C–151 sun-web-app Properties *(Continued)*

Property	Default	Description

valve_*n*	none	This property is deprecated. Use the "valve" on page 242 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 GlassFish Server releases, or the org.glassfish.web.valve.GlassFishValve interface from the current GlassFish Server 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.
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).

TABLE C–151 sun-web-app Properties *(Continued)*

Property	Default	Description
singleThreaded ServletPoolSize	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.
useResponse CTForHeaders	false	If true, response headers are encoded using the response's charset instead of the default (UTF-8).

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 *Oracle GlassFish Server 3.0.1 Application Development Guide*.

Superelements

"entity-mapping" on page 143, "secondary-table" on page 216 (sun-cmp-mappings.xml)

Subelements

none - contains data

target-server

Specifies the IIOP listener for the target server.

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 more information, see "client-container" on page 113.

Superelements

"client-container" on page 113 (sun-acc.xml)

Subelements

The following table describes subelements for the `target-server` element.

TABLE C–152 `target-server` subelements

Element	Required	Description
"`description`" on page 135	zero or one	Specifies the description of the target server.
"`security`" on page 217	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`" on page 247 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

timeout

Specifies the “cache-mapping” on page 107 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” on page 103 element.

Superelements

“cache-mapping” on page 107 (sun-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” on page 151 (sun-ejb-jar.xml)

Subelements

The following table describes subelements for the transport-config element.

TABLE C–155 transport-config Subelements

Element	Required	Description
"integrity" on page 151	only one	Specifies if the target supports integrity-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
"confidentiality" on page 121	only one	Specifies if the target supports privacy-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
"establish-trust-in-target" on page 144	only one	Specifies if the target is capable of authenticating *to* a client. The values are NONE, SUPPORTED, or REQUIRED.
"establish-trust-in-client" on page 144	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" on page 247 (sun-web.xml, sun-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

"sun-application" on page 229 (sun-application.xml), "enterprise-beans" on page 142 (sun-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" on page 107 (sun-web.xml)

Subelements

none - contains data

user-group

Contains the group to which the principal belongs.

Superelements

"security-map" on page 217 (sun-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" on page 136 (sun-ejb-jar.xml)

Subelements

none - contains data

value

Specifies the value of the entity.

Superelements

"call-property" on page 108, "stub-property" on page 228 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml); "property (with subelements)" on page 197 (sun-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

"sun-web-app" on page 232 (sun-web.xml)

Subelements

The following table describes subelements for the valve element.

TABLE C–156 valve Subelements

Element	Required	Description
“description” on page 135	zero or one	Specifies a text description of this element.
“property (with attributes)” on page 195	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 GlassFish Server releases, or the org.glassfish.web.valve.GlassFishValve interface from the current GlassFish Server 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`" on page 153 (`sun-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.

Note – 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`" on page 101 (`sun-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

"sun-application" on page 229 (sun-application.xml)

Subelements

The following table describes subelements for the web element.

TABLE C–158 web Subelements

Element	Required	Description
"web-uri" on page 245	only one	Contains the web URI for the application.
"context-root" on page 129	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" on page 245 (sun-application.xml)

Subelements

none - contains data

webservice-description

Specifies a name and optional publish location for a web service.

Superelements

"sun-web-app" on page 232 (sun-web.xml), "enterprise-beans" on page 142 (sun-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" on page 246	only one	Specifies a unique name for the web service within a web or EJB module.
"wsdl-publish-location" on page 250	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" on page 246 (sun-web.xml, sun-ejb-jar.xml)

Subelements

none - contains data

webservice-endpoint

Specifies information about a web service endpoint.

Superelements

"servlet" on page 221 (sun-web.xml), "ejb" on page 136 (sun-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" on page 192	only one	Specifies a unique name for a port component within a web or EJB module.
"endpoint-address-uri" on page 141	zero or one	Specifies the automatically generated endpoint address.
"login-config" on page 172	zero or one	Specifies the authentication configuration for an EJB web service endpoint.
"message-security-binding" on page 182	zero or one	Specifies a custom authentication provider binding.
"transport-guarantee" on page 240	zero or one	Specifies that the communication between client and server is NONE, INTEGRAL, or CONFIDENTIAL.
"service-qname" on page 220	zero or one	Specifies the WSDL service element that is being referenced.
"tie-class" on page 238	zero or one	Specifies the automatically generated name of a tie implementation class for a port component.
"servlet-impl-class" on page 222	zero or one	Specifies the automatically generated name of the generated servlet implementation class.
"debugging-enabled" on page 133	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)" on page 195 (sun-web.xml) "property (with subelements)" on page 197 (sun-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 GlassFish Server's security domain. It is possible to map multiple EIS group or user principals to the same GlassFish Server principal.

This is different from a "security-map" on page 217, which maps the principal received during servlet or EJB authentication to the credentials accepted by the EIS.

Superelements

"resources" on page 211 (sun-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" on page 135	zero or one	Contains a text description of this element.
"principal-map" on page 194	zero or more	Maps an EIS principal to a principal defined in the GlassFish Server domain.
"group-map" on page 149	zero or more	Maps an EIS group to a group defined in the GlassFish Server 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 Java EE deployment descriptor is used.

Superelements

"service-ref" on page 220 (sun-web.xml, sun-ejb-jar.xml, sun-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" on page 192 (sun-web.xml, sun-ejb-jar.xml, sun-application-client.xml)

Subelements

The following table describes subelements for the wsdl-port element.

TABLE C–163 wsdl-port subelements

Element	Required	Description
"namespaceURI" on page 187	only one	Specifies the namespace URI.

TABLE C–163 wsdl-port subelements *(Continued)*

Element	Required	Description
"localpart" on page 170	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" on page 246 (sun-web.xml, sun-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 sun-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.

Index