Sun Java System Application Server 9.1 Application Deployment Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–3673–10
September 2007

070917@18741

Contents

Preface

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

This preface contains information about and conventions for the entire Sun Java™ System Application Server documentation set.

Application Server Documentation Set

The Application Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for Application Server documentation is `http://docs.sun.com/coll/1343.4`. For an introduction to Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the Application Server Documentation Set

Book Title	Description
Documentation Center	Application Server documentation topics organized by task and subject.
Release Notes	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	How to get started with the Application Server product.
Installation Guide	Installing the software and its components.
Deployment Planning Guide	Evaluating your system needs and enterprise to ensure that you deploy the Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying the server are also discussed.
Application Deployment Guide	Deployment of applications and application components to the Application Server. Includes information about deployment descriptors.

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

Book Title	Description
Developer's Guide	Creating and implementing Java Platform, Enterprise Edition (Java EE platform) applications intended to run on the Application Server that follow the open Java standards model for Java EE components and APIs. Includes information about developer tools, security, debugging, and creating lifecycle modules.
Java EE 5 Tutorial	Using Java EE 5 platform technologies and APIs to develop Java EE applications.
Java WSIT Tutorial	Developing web applications using the Web Service Interoperability Technologies (WSIT). Describes how, when, and why to use the WSIT technologies and the features and options that each technology supports.
Administration Guide	System administration for the Application Server, including configuration, monitoring, security, resource management, and web services management.
High Availability Administration Guide	Post-installation configuration and administration instructions for the high-availability database.
Administration Reference	Editing the Application Server configuration file, `domain.xml`.
Upgrade and Migration Guide	Upgrading from an older version of Application Server or migrating Java EE applications from competitive application servers. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Performance Tuning Guide	Tuning the Application Server to improve performance.
Troubleshooting Guide	Solving Application Server problems.
Error Message Reference	Solving Application Server error messages.
Reference Manual	Utility commands available with the Application Server; written in man page style. Includes the `asadmin` command line interface.

Related Documentation

Application Server can be purchased by itself or as a component of Sun Java Enterprise System (Java ES), a software infrastructure that supports enterprise applications distributed across a network or Internet environment. If you purchased Application Server as a component of Java ES, you should be familiar with the system documentation at `http://docs.sun.com/coll/1286.3`. The URL for all documentation about Java ES and its components is `http://docs.sun.com/prod/entsys.5`.

For documentation about other stand-alone Sun Java System server products, go to the following:

- Message Queue documentation (`http://docs.sun.com/coll/1343.4`)
- Directory Server documentation (`http://docs.sun.com/coll/1224.1`)
- Web Server documentation (`http://docs.sun.com/coll/1308.3`)

A Javadoc™ tool reference for packages provided with the Application Server is located at `http://glassfish.dev.java.net/nonav/javaee5/api/index.html`. Additionally, the following resources might be useful:

- The Java EE 5 Specifications (`http://java.sun.com/javaee/5/javatech.html`)
- The Java EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

For information on creating enterprise applications in the NetBeans™ Integrated Development Environment (IDE), see `http://www.netbeans.org/kb/55/index.html`.

For information about the Java DB database included with the Application 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/`.

Default Paths and File Names

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

TABLE P–2 Default Paths and File Names

Placeholder	Description	Default Value

as-install	Represents the base installation directory for Application Server.	Java ES installations on the Solaris™ operating system:
		`/opt/SUNWappserver/appserver`
		Java ES installations on the Linux operating system:
		`/opt/sun/appserver/`
		Other Solaris and Linux installations, non-root user:
		user's-home-directory`/SUNWappserver`
		Other Solaris and Linux installations, root user:
		`/opt/SUNWappserver`
		Windows, all installations:
		SystemDrive`:\Sun\AppServer`

TABLE P–2 Default Paths and File Names *(Continued)*

Placeholder	Description	Default Value

domain-root-dir	Represents the directory containing all domains.	Java ES Solaris installations:
		`/var/opt/SUNWappserver/domains/`
		Java ES Linux installations:
		`/var/opt/sun/appserver/domains/`
		All other installations:
		as-install`/domains/`

domain-dir	Represents the directory for a domain. In configuration files, you might see *domain-dir* represented as follows: `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-dir*
instance-dir	Represents the directory for a server instance.	*domain-dir*`/`*instance-dir*

Typographic Conventions

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

TABLE P–3 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.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–4 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.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Documentation, Support, and Training

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

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Searching Sun Product Documentation

Besides searching Sun 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 Sun 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 – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun 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. Sun 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.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-3673.

◆ ◆ ◆ CHAPTER 1

Assembling and Deploying Applications

This chapter describes Sun Java™ System Application Server modules and how these modules are assembled separately or together in an application. This chapter also describes tools for assembly and deployment.

The Application Server modules and applications include Java Platform, Enterprise Edition (Java EE platform) standard features and Application Server specific features. Only Application Server specific features are described in detail in this chapter.

The following topics are presented in this chapter:

Note – Some topics in the documentation pertain to features that are available only in domains that are configured to support clusters. Examples of domains that support clusters are domains that are created with the cluster profile or the enterprise profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*.

Overview of Assembly and Deployment

Application assembly (also known as packaging) is the process of combining discrete components of an application into a single unit that can be deployed to a Java-EE-compliant application server. A package can be classified either as a module or as a full-fledged application. This section covers the following topics:

About Modules

A Java EE module is a collection of one or more Java EE components that execute in the same container type (for example, web or EJB) with annotations or deployment descriptors of that type. One descriptor is Java EE standard, the other is optional and Application Server specific. Annotations can be used instead of Java EE standard descriptors.

Types of Java EE modules are as follows:

- **Web Application Archive (WAR)** — A web application is a collection of servlets, HTML pages, classes, and other resources that can be bundled and deployed to several Java EE application servers. 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, and annotations or deployment descriptors (`web.xml` and `sun-web.xml`).
- **EJB JAR File** — The EJB JAR file is the standard format for assembling enterprise beans. This 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, a `.dbschema` file and a CMP mapping descriptor, `sun-cmp-mapping.xml`, can be included as well.
- **Application Client JAR File** — An application client is an Application Server specific type of Java EE client. An application client supports the standard Java EE Application Client specifications, and in addition, supports direct access to the Application Server. Its deployment descriptors are `application-client.xml` and `sun-application-client.xml`.
- **Resource RAR File** — RAR files apply to J2EE CA connectors. A connector extends the EJB container to allow access to external systems much like a device driver provides access to a peripheral device from a process hosted by an operating system. It is a portable way of allowing EJB components to access a foreign enterprise system. Each Application Server connector has annotations or a Java EE XML file, `ra.xml`.

Package definitions must be used in the source code of all modules so the class loader can properly locate the classes after the modules have been deployed.

Because the information in a deployment descriptor is declarative, it can be changed without requiring modifications to source code. At run time, the Java EE server reads this information and acts accordingly.

The Application Server also supports lifecycle modules. See Chapter 13, "Developing Lifecycle Listeners," in *Sun Java System Application Server 9.1 Developer's Guide* for more information.

EJB JAR, Web, and application client modules can also be deployed separately, outside of any application, as in the following figure. EJB components are assembled in a JAR file with annotations or `ejb-jar.xml` and `sun-ejb-jar.xml` deployment descriptors. Web components are assembled in a WAR file with annotations or `web.xml` and `sun-web.xml` deployment descriptors. Application client components are assembled in a JAR file with `application-client.xml` and `sun-application-client.xml` deployment descriptors. These module types are deployed to the Application Server.

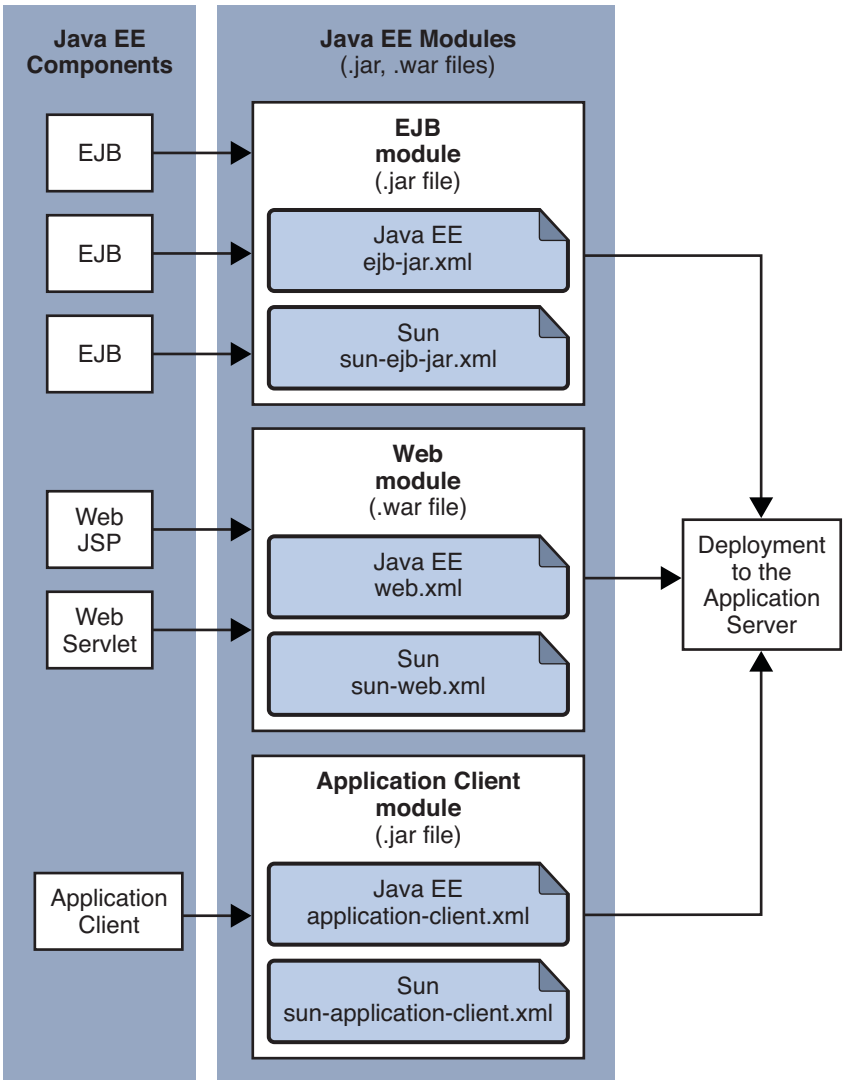

FIGURE 1–1 Module Assembly and Deployment

About Applications

A Java EE application is a logical collection of one or more Java EE modules tied together by application annotations or deployment descriptors. Components can be assembled at either the module or the application level. Components can also be deployed at either the module or the application level.

The following diagram illustrates how components are assembled into modules and then assembled into an Application Server application and deployed. EJB components are assembled in a JAR file with annotations or `ejb-jar.xml` and `sun-ejb-jar.xml` deployment descriptors. Web components are assembled in a WAR file with annotations or `web.xml` and `sun-web.xml` deployment descriptors. An application client is assembled in a JAR file with `application-client.xml` and `sun-application-client.xml` deployment descriptors. A resource adapter is assembled in a RAR file with a `ra.xml` deployment descriptor. All modules are assembled in an EAR file and deployed to the Application Server.

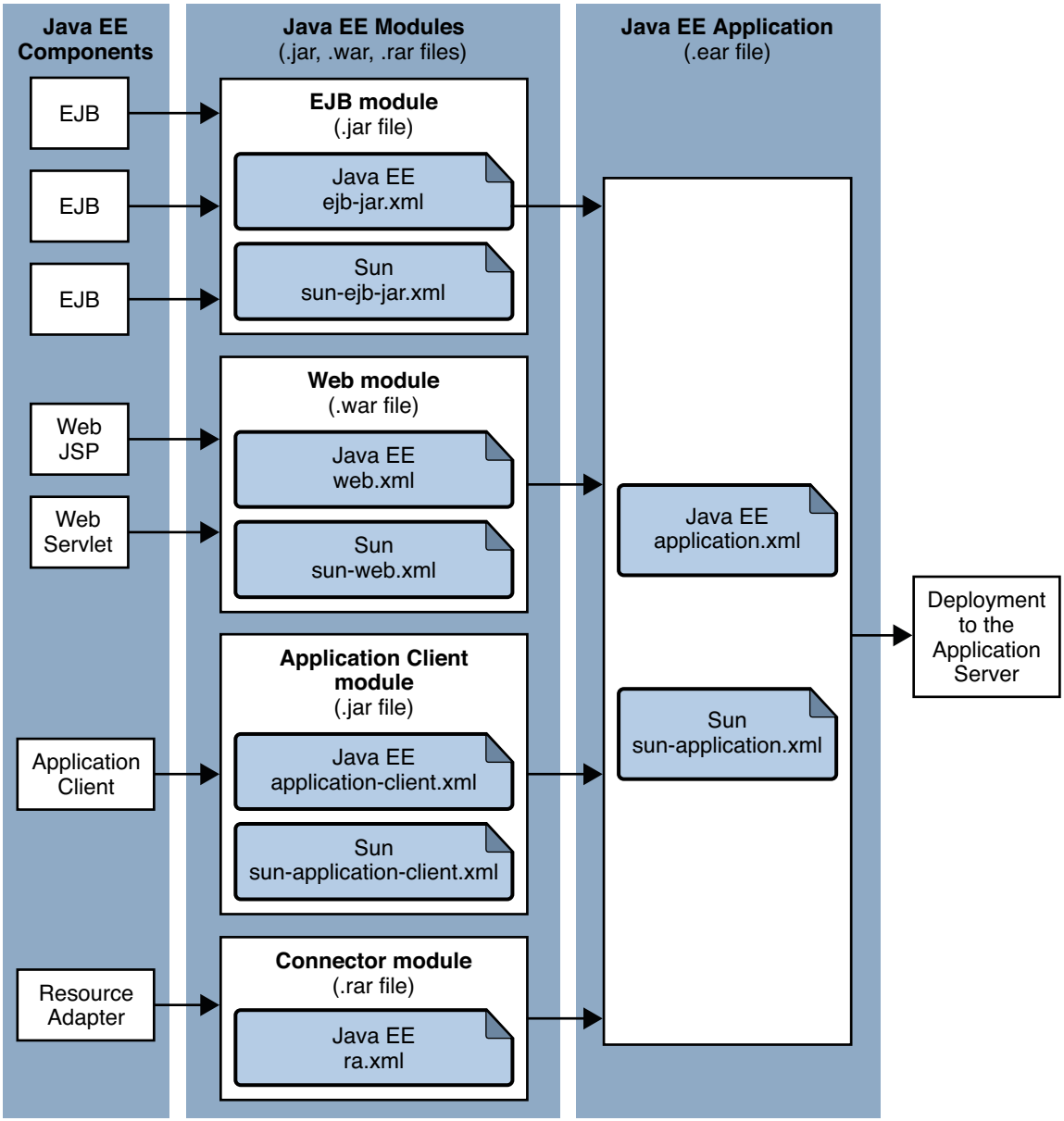

FIGURE 1–2 Application Assembly and Deployment

Each module has an optional Application Server deployment descriptor and annotations or a Java EE deployment descriptor. The Application Server uses the annotations or deployment descriptors to deploy the application components and to register the resources with the Application Server.

An application consists of one or more modules, an optional Application Server deployment descriptor, and annotations or a Java EE application deployment descriptor. All items are assembled, using the Java ARchive (`.jar`) file format, into one file with an extension of `.ear`.

Java EE Standard Annotation

The Application Server supports applications and modules annotated 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 3.0 Specification (`http://www.jcp.org/en/jsr/detail?id=220`)

The following annotation and deployment descriptor combinations are supported:

- Java EE applications or modules can be packaged with full Java EE 5.0 compliant standard and runtime deployment descriptors. If the standard deployment descriptors have specified the attribute `metadata-complete`, annotations in the application or 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 Application Server deployment descriptors are also optional.
- 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, but a warning message is logged.

To check the correctness of annotations or deployment descriptors prior to deployment, see "The `verifier` Utility" on page 26.

Java EE Standard Descriptors

Java EE standard deployment descriptors are described in the Java EE specification, v5. You can find the specification at `http://java.sun.com/products/`. Information about the XML schemas that define Java EE standard deployment descriptors is available at `http://java.sun.com/xml/ns/javaee/`.

Application Server Descriptors

The Application Server uses additional, optional deployment descriptors for configuring features specific to the Application Server.

To check the correctness of these deployment descriptors prior to deployment, see "The `verifier` Utility" on page 26.

For complete descriptions of these files, see Appendix A, "Deployment Descriptor Files."

Naming Standards

Names of applications and individually deployed EJB JAR, WAR, and connector RAR modules must be unique within an Application Server domain. Modules of the same type within an application must have unique names. In addition, for entity beans that use CMP, `.dbschema` file names must be unique within an application.

If you do not explicitly specify a name, the default name is the first portion of the file name (without the `.war` or `.jar` extension). Modules of different types can have the same name within an application, because the directories holding the individual modules are named with `_jar`, `_war` and `_rar` suffixes. This is the case when you use the Admin Console or the `asadmin` command. See "Tools for Deployment" on page 37.

You can specify a name in one of these ways:

- If deploying using the Admin Console, specify the name in the Application Name field.
- If deploying using the `asadmin deploy` command, the default name of the application or module is the prefix of the JAR file that you are deploying. For example, for the `hello.war` file, the Web application name is `hello`. To override the default name, specify the `--name` option.

Make sure your package and file names do not contain spaces or characters that are illegal for your operating system.

Using a Java package-like naming scheme is recommended for module filenames, EAR filenames, 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. The use of this package-like naming scheme ensures that name collisions do not occur. The benefits of this naming practice apply not only to the Application Server, but to other Java EE application servers as well.

JNDI lookup names for EJB components must also be unique. Establishing a consistent naming convention might help. For example, appending the application name and the module name to the EJB name is one way to guarantee unique names. In this case, `mycompany.pkging.pkgingEJB.MyEJB` would be the JNDI name for an EJB in the module `pkgingEJB.jar`, which is packaged in the application `pkging.ear`.

If you are writing your own JSR 88 client to deploy applications to the Application Server using the following API, the name of the application is taken from the `display-name` entry in the Java EE standard deployment descriptor, because there is no file name in this case. If the `display-name` entry is not present, the Application Server creates a temporary file name and uses that name to deploy the application.

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

Neither the Admin Console nor the `asadmin` command uses this API.

Note – Use of the following JSR 88 API is preferred. In this case, the name is derived from the file name as previously described.

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

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

Directory Structure

When you deploy an application, the application is expanded from the EAR file to an open directory structure. The directories holding the individual modules are named with `_jar`, `_war` and `_rar` suffixes. If you use the `asadmin deploydir` command to deploy a directory instead of an EAR file, your directory structure must follow this same convention. Module and application directory structures follow the structure outlined in the Java EE specification.

Assembling Modules and Applications

Assembling (or packaging) modules and applications in Application Server conforms to all of the customary Java-EE-defined specifications. The only difference is that when you assemble in Application Server, you can include optional Application Server specific deployment descriptors that enhance the functionality of the Application Server. You can also package the Application Server specific deployment descriptors separately in a deployment plan; see "Using a Deployment Plan" on page 36.

For example, when you assemble an EJB JAR module, you annotate or create two deployment descriptor files with these names: `ejb-jar.xml` and `sun-ejb-jar.xml`. If the EJB component is an entity bean with container-managed persistence, you can also create a `.dbschema` file and a `sun-cmp-mapping.xml` file. For more information about `sun-ejb-jar.xml` and `sun-cmp-mapping.xml`, see Appendix A, "Deployment Descriptor Files."

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, package the EJB module and utility JAR within an application using the JAR Extension Mechanism Architecture. For other alternatives, see Chapter 2, "Class Loaders," in *Sun Java System Application Server 9.1 Developer's Guide*.

The Application Server provides these tools for assembling and verifying a module or an application:

- "The `asant` Utility" on page 26
- "The NetBeans IDE" on page 26
- "The `verifier` Utility" on page 26

The `asant` Utility

The `asant` utility can help you assemble and deploy modules and applications. For details, see Chapter 3, "The asant Utility," in *Sun Java System Application Server 9.1 Developer's Guide*.

The NetBeans IDE

You can use the NetBeans™ Integrated Development Environment (IDE) to assemble Java EE applications and modules. The GlassFish edition of the Application Server is bundled with the NetBeans 5.5 IDE. For more information about using the NetBeans IDE, see `http://www.netbeans.org`.

The `verifier` Utility

The `verifier` utility validates Java EE annotations and deployment descriptors and Application Server specific deployment descriptors against their corresponding DTD or schema files. It gives errors and warnings if a module or application is not Java EE and Application Server compliant. You can verify deployment descriptors in EAR, WAR, RAR, and JAR files.

The `verifier` tool is not simply an XML syntax verifier. Rules and interdependencies between various elements in the deployment descriptors are verified. Where needed, user application classes are introspected to apply validation rules.

The `verifier` is integrated into Application Server deployment and the `asant` task "The sun-appserv-deploy Task" in *Sun Java System Application Server 9.1 Developer's Guide*. You can also run it as a stand-alone utility from the command line. The `verifier` utility is located in the *as-install*`/bin` directory.

You can run the `verifier` during Application Server deployment using the Admin Console or the `asadmin deploy` command with the `--verify="true"` option (see "The `asadmin` Deployment Commands" on page 38 and "The Admin Console Deployment Pages" on page 38). In these cases, the output of the `verifier` is written to the *tempdir*`/verifier-results/` directory, where *tempdir* is the temporary directory defined in the operating system. Deployment fails if any failures are reported during the verification process. The `verifier` also logs information about verification progress to the standard output.

For details on all the assertions tested by the `verifier`, see the assertions documentation provided at `http://java.sun.com/j2ee/avk/index.html`.

Tip – Using the `verifier` tool can help you avoid runtime errors that are difficult to debug.

This section covers the following topics:

Command-Line Syntax

The `verifier` tool's syntax is as follows:

`verifier [`*options*`]` *file*

The *file* can be an EAR, WAR, RAR, or JAR file.

The following table shows the *options* for the `verifier` tool.

TABLE 1–1 Verifier Options

Short Form	Long Form	Description
`-v`	`--verbose`	Turns on verbose mode. In verbose mode, the status of each run of each test is displayed on the verifier console.
`-d` *output-dir*	`--destdir` *output-dir*	Writes test results to the *output-dir*, which must already exist. By default, the results files are created in the current directory.
`-D` *domain-dir*	`--domain` *domain-dir*	Specifies the absolute path to the domain directory. This option is ignored if the `-p` option is used. The default domain directory is*as-install*`/domains/domain1`.
`-r`*level*	`--reportlevel` *level*	Sets the output report *level* to one of the following values: ■ `a` or `all` - Reports all results. ■ `w` or `warnings` - Reports only warning and failure results. This is the default. ■ `f` or `failures` - Reports only failure results.

TABLE 1–1 Verifier Options *(Continued)*

Short Form	Long Form	Description
`-t`	`--timestamp`	Appends a timestamp to the output file name. The format of the timestamp is yyyyMMddhhmmss.
`-?`	`--help`	Displays help for the `verifier` command. If you use this option, you do not need to specify an EAR, WAR, RAR, or JAR file.
`-V`	`--version`	Displays the `verifier` tool version. If you use this option, you do not need to specify an EAR, WAR, RAR, or JAR file.
`-p`	`--portability`	Verifies portable features only. By default, the `verifier` also checks correct usage of Application Server features in the `sun-*.xml` deployment descriptor files.
`-a`	`--app`	Runs only the application tests.
`-A`	`--appclient`	Runs only the application client tests.
`-c`	`--connector`	Runs only the connector tests.
`-e`	`--ejb`	Runs only the EJB tests.
`-w`	`--web`	Runs only the web module tests.
`-s`	`--webservices`	Runs only the web service tests.
`-l`	`--webservicesclient`	Runs only the web service client tests.

For example, the following command runs the `verifier` on the `ejb.jar` file with default settings:

```
verifier ejb.jar
```

The results files are `ejb.jar.txt` and `ejb.jar.xml`.

For a more complex example, the following command runs the `verifier` on the `ejb.jar` file in portability mode displaying only failures and with a timestamp:

```
verifier -p -rf -t ejb.jar
```

The results files are `ejb.jar`*timestamp*`.txt` and `ejb.jar`*timestamp*`.xml`. The format of the *timestamp* is yyyyMMddhhmmss.

If the `verifier` runs successfully and no verification errors occurred, a result code of `0` is returned. A nonzero error code is returned if errors occurred or the `verifier` fails to run.

Ant Integration

You can integrate the `verifier` into an Ant build file as a target and use the Ant call feature to call the target each time an application or module is assembled. You can use any of the arguments described in Table 1–1. Example code for an Ant verify target is as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<project name="Verifier Launcher" default="verify">
    <target name="verify" description="verify using verifier script">
        <exec executable="as-install/bin/verifier"
                failonerror="true" vmlauncher="false">
            <arg line="-d /tmp path-to-app"/>
        </exec>
    </target>
</project>
```

Sample Results Files

Here is a sample results XML file:

```
<static-verification>
 <ejb>
     <failed>
         <test>
             <test-name>
tests.ejb.session.TransactionTypeNullForContainerTX
            </test-name>
             <test-assertion>
Session bean with bean managed transaction demarcation test
            </test-assertion>
             <test-description>
For [ TheGreeter ] Error: Session Beans [ TheGreeter ] with
[ Bean ] managed transaction demarcation should not have
container transactions defined.
             </test-description>
         </test>
     </failed>
 </ejb>
...
</static-verification>
```

Here is a sample results TXT file:

```
---------------------------
STATIC VERIFICATION RESULTS
---------------------------


---------------------------------
NUMBER OF FAILURES/WARNINGS/ERRORS
---------------------------------
# of Failures : 1
# of Warnings : 0
# of Errors : 0
```

```
-----------------------------
RESULTS FOR EJB-RELATED TESTS
-----------------------------

-------------
FAILED TESTS :
-------------

Test Name : tests.ejb.ejb30.BusinessIntfInheritance
Test Assertion : A business interface must not extend javax.ejb.EJBObject
or javax.ejb.EJBLocalObject. Please refer to EJB 3.0 Simplified API Section
#3.2 for further information.
Test Description : For [ sessionApp#session-ejb.jar#HelloEJB ]
[ com.sun.sample.session.Hello ] extends either javax.ejb.EJBObject or
javax.ejb.EJBLocalObject.


---------------------------------
END OF STATIC VERIFICATION RESULTS
---------------------------------
```

Deploying Modules and Applications

This section describes the different ways to deploy Java EE applications and modules to the Application Server. It covers the following topics:

Deployment Errors

If an error occurs during deployment, the application or module is not deployed. If a module within an application contains an error, the entire application is not deployed. This prevents a partial deployment that could leave the server in an inconsistent state.

In addition, certain warning conditions allow an application to be deployed but return a warning message to the deployment client.

The Deployment Life Cycle

After installing the Application Server and starting a domain, you can deploy (install) Java EE applications and modules. During deployment and as the application is changed, an application or module can go through the following stages:

1. Initial Deployment

   Before deploying an application or module, start the domain.

   Deploy (install) an application or module to a specific stand-alone server instance or cluster. Because applications and modules are packaged in archive files, specify the archive file name during deployment. The default is to deploy to the default server instance `server`.

   If you deploy to server instances or clusters, the application or module exists in the domain's central repository and is referenced by any clusters or server instances you deployed to as targets.

   You can also deploy to the domain using the `asadmin deploy` or `asadmin deploydir` command or the Admin Console. In the cluster profile, the preselected deployment target is `server`. To deploy only to the domain, specify no target. If you deploy the application or module only to the domain, it exists in the domain's central repository, but no server instances or clusters reference it until you add references.

   Deployment is *dynamic*: you don't need to restart the server instance after deploying application or module for applications to be available. If you do restart, all deployed applications and modules are still deployed and available.

2. Enabling or Disabling

   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 or cluster. To prevent access, disable the application or module. A disabled application or module is not uninstalled from the domain and can be easily enabled after deployment. For more information, see "Disabling a Deployed Application or Module" on page 33.

3. Adding or Deleting Targets for a Deployed Application or Module

   Once deployed, the application or module exists in the central repository and can be referenced by a number of server instances and/or clusters. Initially, the server instances or clusters you deployed to as targets reference the application or module.

   To change which server instances and clusters reference an application or module after it is deployed, change an application or module's targets using the Admin Console, or change the application references using the `asadmin` tool. Because the application itself is stored in the central repository, adding or deleting targets adds or deletes the same version of an application on different targets. However, an application deployed to more than one target

can be enabled on one and disabled on another, so even if an application is referenced by a target, it is not available to users unless it is enabled on that target.

4. Redeployment

   To replace a deployed application or module, redeploy it. Redeploying automatically undeploys the previously deployed application or module and replaces it with the new one.

   When redeploying through the Admin Console, the redeployed application or module is deployed to the domain, and all stand-alone or clustered server instances that reference it automatically receive the new version if dynamic reconfiguration is enabled. If using the `asadmin deploy` command to redeploy, specify `domain` as the target.

   If an application is deployed on one target only and you are redeploying using `asadmin`, specify the same target during redeployment.

   For production environments, perform a rolling upgrade, which upgrades the application or module without interruption in service. For more information, see "To Upgrade Components Without Loss of Service" in *Sun Java System Application Server 9.1 High Availability Administration Guide*.

5. Undeployment

   To uninstall an application or module, undeploy it.

Deployment for Development

This section covers the following topics related to deployment for development:

Note – You can overwrite a previously deployed application by using the `--force` option of `asadmin deploy` or `asadmin deploydir` or by selecting the Redeploy button in the Admin Console. However, you must remove a preconfigured resource before you can update it.

Dynamic Deployment

You can deploy, redeploy, and undeploy an application or module without restarting the server instances. This is called dynamic deployment. Although primarily for developers, dynamic deployment can be used in operational environments to bring new applications and modules online without requiring a server restart.

Whenever a redeployment is done, the sessions at that transit time become invalid. The client must restart the session.

Disabling a Deployed Application or Module

You can disable a deployed application or module without removing it from the server. Disabling an application makes it inaccessible to clients.

To disable an application or module using the `asadmin disable` command, see the *Sun Java System Application Server 9.1 Reference Manual*.

▼ To Disable an Application or Module in the Admin Console

1. **Open the Applications component.**
2. **Go to the page for the type of application or module.**
   For example, for a web application, go to the Web Applications page.
3. **Click on the box to the left of the name of each application or module you wish to disable.**
4. **Click on the Disable button.**

See Also For details, click the Help button in the Admin Console.

Dynamic Reloading

If dynamic reloading is enabled (it is by default), you do not have to redeploy an application or module when you change its code or deployment descriptors. Simply copy the changed JSP or class files into the deployment directory for the application or module. The server checks for changes periodically and redeploys the application, automatically and dynamically, with the changes. This feature is available only on the default server instance. Deployment directories are as follows:

For an application:

domain-dir`/applications/j2ee-apps/`*app-name*

For an individually deployed module:

domain-dir`/applications/j2ee-modules/`*module-name*

Dynamic reloading is useful in a development environment, because it allows code changes to be tested quickly. In a production environment, however, dynamic reloading might degrade performance. In addition, whenever a reload is done, the sessions at that transit time become invalid. The client must restart the session.

You can also use the `sun-appserv-update` task with the `asant` utility to update an application or module. See Chapter 3, "The asant Utility," in *Sun Java System Application Server 9.1 Developer's Guide*.

To Enable Dynamic Reloading in the Admin Console

1. **To access reloading features:**
   - **In the developer profile:**
     a. **Select the Application Server component.**
     b. **Select the Advanced tab.**
     c. **Select the Applications Configuration tab.**
   - **In the cluster profile:**
     a. **Select the Stand-Alone Instances component.**
     b. **Select the instance named `server` in the table.**
        This is the Admin Server.
     c. **Select the Advanced tab.**
2. **Check the Reload Enabled box to enable dynamic reloading.**
3. **Enter a number of seconds in the Reload Poll Interval field.**
   This sets the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.

See Also For details, click the Help button in the Admin Console.

To Reload Code or Deployment Descriptor Changes

1. **Create an empty file named `.reload` at the root of the deployed application or module.**
   For an application:
   domain-dir`/applications/j2ee-apps/`*app-name*`/.reload`
   For an individually deployed module:
   domain-dir`/applications/j2ee-modules/`*module-name*`/.reload`
2. **Explicitly update the `.reload` file's timestamp (`touch .reload` in UNIX) each time you make changes.**

Automatic Deployment

Automatic deployment, also called *autodeployment*, involves copying an application or module file (JAR, WAR, RAR, or EAR) into a special directory, where it is automatically deployed by the Application Server. To undeploy an automatically deployed application or module, simply remove its file from the special autodeployment directory. This is useful in a development environment, because it allows new code to be tested quickly. This feature is available only on the default server instance.

Autodeployment of a web services JSR 181 annotated file is supported. For more information, see JSR 181 (`http://www.jcp.org/en/jsr/detail?id=181`) and Chapter 6, "Developing Web Services," in *Sun Java System Application Server 9.1 Developer's Guide*.

Autodeployment is enabled by default.

▼ To Enable and Configure or to Disable Autodeployment

1. **To access autodeployment features:**
   - **In the developer profile:**
     a. **Select the Application Server component.**
     b. **Select the Advanced tab.**
     c. **Select the Applications Configuration tab.**
   - **In the cluster profile:**
     a. **Select the Stand-Alone Instances component.**
     b. **Select the instance named `server` in the table.**
        This is the Admin Server.
     c. **Select the Advanced tab.**
2. **Check the Auto Deploy Enabled box to enable autodeployment, or uncheck this box to disable autodeployment.**
3. **Enter a number of seconds in the Auto Deploy Poll Interval field.**
   This sets the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.

4 **You can change the Auto Deploy Directory.**

You can enter an absolute or relative path. A relative path is relative to *domain-dir*. The default is *domain-dir*`/autodeploy`.

5 **You can check the Verifier Enabled box to verify your deployment descriptor files. This is optional.**

For details about the `verifier`, see "The `verifier` Utility" on page 26.

6 **Check the Precompile Enabled box to precompile any JSP files.**

See Also For details, click the Help button in the Admin Console.

Directory Deployment

A directory containing an unpackaged application or module is sometimes called an exploded directory. To deploy a directory instead of an EAR, WAR, JAR, or RAR file, you can do one of the following:

- Use the Admin Console as described in "The Admin Console Deployment Pages" on page 38 and enter a path to an exploded directory instead of a path to an archive file.
- Use the `asadmin deploydir` command. See the *Sun Java System Application Server 9.1 Reference Manual*. The `asadmin deploydir` command is available only on the default server instance.

The contents of the directory must match the contents of a corresponding Java EE archive file. For example, if you deploy a Web application from a directory, the contents of the directory must be the same as a corresponding WAR file. In addition, the directories holding the individual modules must be named with `_jar`, `_war` and `_rar` suffixes. For information about the required directory contents, see the appropriate specifications.

You can change the deployment descriptor files directly in the exploded directory. If your environment is configured to use dynamic reloading, you can also dynamically reload applications deployed from the directory. For more information, see "Dynamic Reloading" on page 33.

Note – On Windows, if you are deploying a directory on a mapped drive, you must be running the Application Server as the same user to which the mapped drive is assigned, or the Application Server won't see the directory.

Using a Deployment Plan

This feature is for advanced developers.

A deployment plan is an JAR file that contains only the deployment descriptors that are specific to the Application Server. The deployment plan is part of the implementation of JSR 88. Use a deployment plan to deploy an application or module that does not contain the deployment descriptors that are specific to the Application Server. For more information about JSR 88, see the JSR 88 page at `http://jcp.org/en/jsr/detail?id=88`.

To deploy using a deployment plan, specify the `--deploymentplan` option of the `asadmin deploy` command. The following command, for example, deploys the enterprise application in the `myrostapp.ear` file according to the plan specified by the `mydeployplan.jar` file.

```
$ asadmin deploy --user admin --deploymentplan mydeployplan.jar myrostapp.ear
```

In the deployment plan file for an enterprise application (EAR), the `sun-application.xml` file is located at the root. The deployment descriptor for each module is stored according to this syntax: *module-name.sun-dd-name*, where the *sun-dd-name* depends on the module type. If a module contains a CMP mappings file, the file is named *module-name*`.sun-cmp-mappings.xml`. A `.dbschema` file is stored at the root level with each forward slash character (/) replaced by a pound sign (#). The following listing shows the structure of the deployment plan file for an enterprise application (EAR).

```
$ 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
```

In the deployment plan for a web application or a module file, the deployment descriptor that is specific to the Application Server is at the root level. If a stand-alone 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
```

Tools for Deployment

This section discusses the various tools that can be used to deploy modules and applications. The deployment tools include:

- "The `asant` Utility" on page 38
- "JSR 88" on page 38

The asant Utility

The asant utility can help you assemble and deploy modules and applications. For details, see Chapter 3, "The asant Utility," in *Sun Java System Application Server 9.1 Developer's Guide*.

JSR 88

You can write your own JSR 88 client to deploy applications to the Application Server. For more information, see the JSR 88 page at http://jcp.org/en/jsr/detail?id=88.

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

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

The :https is not needed by default in the developer profile and needed by default in the cluster profile. For example:

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

See "Naming Standards" on page 24 for application and module naming considerations.

The asadmin Deployment Commands

You can use the asadmin deploy or asadmin deploydir command to deploy or undeploy applications and individually deployed modules on local servers. For details, see "Directory Deployment" on page 36 and the *Sun Java System Application Server 9.1 Reference Manual*. The asadmin deploydir command is available only on the default server instance.

To deploy a lifecycle module, see "Deploying a Lifecycle Module" on page 41.

The Admin Console Deployment Pages

You can use the Admin Console to deploy modules and applications to both local and remote Application Server sites.

▼ To Use the Admin Console for Deployment

1 Open the Applications component.

2 Go to the page for the type of application or module.

For example, for a web application, go to the Web Applications page. In the developer profile, you can undeploy, enable, or disable an application or module from the table on this page. In the cluster profile, click on the link in the Status column to undeploy, enable, or disable an application or module on specific targets.

3 **Click on the Deploy button.**

On this page, you type the path to the WAR, JAR, EAR, or RAR file or the exploded directory structure. You can also specify deployment settings that vary according to the type of application or module.

See Also For details, click the Help button in the Admin Console.

To deploy a lifecycle module, see "Deploying a Lifecycle Module" on page 41.

For more information on deploying from an exploded directory structure, see "Directory Deployment" on page 36.

Deployment by Module or Application

You can deploy applications or individual modules that are independent of applications; see "About Modules" on page 18 and "About Applications" on page 20. Individual module-based deployment is preferable when components need to be accessed by:

- Other modules
- Java EE applications
- Application clients (Module-based deployment allows shared access to a bean from an application client, a servlet, or an EJB component.)

Modules can be combined into an EAR file and then deployed as a single module. This is similar to deploying the modules of the EAR independently.

Deploying a Web Service

You deploy a web service endpoint to the Application Server just as you would any servlet or stateless session bean (SLSB).

Web service management is fully supported in the Admin Console. If the deployed application or module has a web service endpoint, it is detected automatically during deployment. Once the application or module is deployed, click on the Web Service component. The table in the right frame lists deployed web service endpoints.

You can use the `--registryjndiname` option of the `asadmin deploy` or `asadmin deploydir` command to publish the web service as part of deployment, but this is optional. See "Tools for Deployment" on page 37.

To deploy a JSR 181 annotated file, use the autodeployment feature. You can compile and deploy in one step, as in the following example:

```
javac -cp javaee.jar -d domain-dir/autodeploy MyWS.java
```

For more information about JSR 181, see `http://jcp.org/en/jsr/detail?id=181`. For more information about autodeployment, see "Automatic Deployment" on page 35.

The Sun-specific deployment descriptor files `sun-web.xml` and `sun-ejb-jar.xml` provide optional web service enhancements in their "webservice-endpoint" on page 176 and "webservice-description" on page 175 elements.

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 *Sun Java System Application Server 9.1 Developer's Guide*.

Deploying a WAR Module

You deploy a WAR module as described in "Tools for Deployment" on page 37. If you do not specify a context root, the default is the name of the WAR file without the extension.

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 used to look up the resource.

You can precompile JSP files during deployment by checking the appropriate box in the Admin Console, or by using the `--precompilejsp` option of the `asadmin deploy` or `asadmin deploydir` command. The `asant` tasks `sun-appserv-deploy` and `sun-appserv-jspc` also allow you to precompile JSP files. For more information, see Chapter 3, "The asant Utility," in *Sun Java System Application Server 9.1 Developer's Guide*.

You can keep the generated source for JSP files by adding the `-keepgenerated` flag to the `jsp-config` element in `sun-web.xml`. If you include this property when you deploy the WAR module, the generated source is kept in *domain-dir*`/generated/jsp/j2ee-apps/`*app-name*`/`*module-name*, if it is in an application, or *domain-dir*`/generated/jsp/j2ee-modules/`*module-name*, if it is in an individually deployed web module.

For more information about JSP precompilation, see "jsp-config" on page 108 and "Options for Compiling JSP Files" in *Sun Java System Application Server 9.1 Developer's Guide*.

HTTP sessions in WAR modules can be saved in a persistent store in case a server instance fails. For more information, see "Distributed Sessions and Persistence" in *Sun Java System Application Server 9.1 Developer's Guide* and the *Sun Java System Application Server 9.1 High Availability Administration Guide*.

Note – After a web application is undeployed, its `HttpSession` information is not immediately removed if sessions are persistent. `HttpSession` information is removed in the subsequent cycle, when timed out sessions are removed. Therefore, you should disable a web application before undeploying it if sessions are persistent.

Web module context roots must be unique within a server instance.

See the *Sun Java System Application Server 9.1 High Availability Administration Guide* for information about load balancing.

Deploying an EJB JAR Module

You deploy an EJB JAR module as described in "Tools for Deployment" on page 37.

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.

You can keep the generated source for stubs and ties by adding the `-keepgenerated` flag to the `rmic-options` attribute of the `java-config` element in `domain.xml`. If you include this flag when you deploy the EJB JAR module, the generated source is kept in *domain-dir*`/generated/ejb/j2ee-apps/`*app-name*`/`*module-name*, if it is in an application, or *domain-dir*`/generated/ejb/j2ee-modules/`*module-name*, if it is in an individually deployed EJB JAR module. For more information about the `-keepgenerated` flag, see the *Sun Java System Application Server 9.1 Administration Reference*.

Generation of stubs and ties is performed asynchronously, so unless you request their generation during deployment, stubs and ties are not guaranteed to be available immediately after deployment. To generate stubs and ties during deployment, use the `--retrieve` option of the `asadmin deploy` or `asadmin deploydir` command, or check the Generate RMIStubs Enabled box in the Admin Console.

You can use the `asadmin get-client-stubs` command to retrieve the stubs and ties whether or not you requested their generation during deployment. For details, see the *Sun Java System Application Server 9.1 Reference Manual*.

Deploying a Lifecycle Module

For general information about lifecycle modules, see Chapter 13, "Developing Lifecycle Listeners," in *Sun Java System Application Server 9.1 Developer's Guide*.

You can deploy a lifecycle module using the following tools:

- In the Admin Console, open the Applications component and go to the Lifecycle Modules page. For details, click the Help button in the Admin Console.
- Use the `asadmin create-lifecycle-module` command. For details, see the *Sun Java System Application Server 9.1 Reference Manual*.

Note – If the `is-failure-fatal` setting is set to `true` (the default is `false`), lifecycle module failure prevents server initialization or startup, but not shutdown or termination.

Deploying an Application Client

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. It is enabled by default both in application clients and in the Application Server.

This section contains the following topics:

▼ To Deploy an Application Client

1. **Assemble the necessary client files.**
2. **Assemble the EJB components to be accessed by the client.**
3. **Package the client and EJB components together in an application.**
4. **Deploy the application as described in "Tools for Deployment" on page 37.**
5. **If you are using the `appclient` script to run the application client, retrieve the client JAR file.**

   The client JAR file contains the ties and necessary classes for the application client.

   You can use the `--retrieve` option to get the client JAR file.

   You can also use the `asadmin get-client-stubs` command to retrieve the stubs and ties whether or not you requested their generation during deployment. For details, see the *Sun Java System Application Server 9.1 Reference Manual*.
6. **If you are using the `appclient` script to run the application client, copy the client JAR file to the client machine.**

Next Steps You can execute the client on the Application Server machine to test it 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 page to launch the application client using Java Web Start.
- You can also use the `appclient` script in the *as-install*`/bin` directory to run an application client. 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 *Sun Java System Application Server 9.1 Developer's Guide*.

For more detailed information about the `appclient` script, see the *Sun Java System Application Server 9.1 Reference Manual*.

▼ To Prepare Another Machine for Executing an Application Client

If Java Web Start is enabled, the default URL for an application is as follows:

`http://`*host*`:`*port*`/`*context-root*

The default URL for a stand-alone application client module is as follows:

`http://`*host*`:`*port*`/`*module-id*

If the *context-root* or *module-id* is not specified during deployment, the name of the EAR or JAR file without the 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 24.

To set a different URL for an application client, use the `context-root` subelement of the "java-web-start-access" on page 106 element in the `sun-application-client.xml` file.

If Java Web Start is *not* enabled for the application client, follow these steps.

1. **You can use the `package-appclient` script in the *as-install*`/bin` directory to create the ACC package JAR file. This is optional.**

   This JAR file is created in the *as-install*`/lib/appclient` directory.

2. **Copy the ACC package JAR file to the client machine and unjar it.**

3. **Configure the `sun-acc.xml` file.**

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

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

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

5. **Copy the client JAR file to the client machine.**

   You are now ready to execute the client.

See Also For more detailed information about Java Web Start and the `package-appclient` script, see Chapter 11, "Developing Java Clients," in *Sun Java System Application Server 9.1 Developer's Guide*.

Undeployment of Application Clients

Once application clients are downloaded, they remain on the client until removed manually. The Java Web Start control panel provides a simple interface you can use to discard downloaded application clients that used Java Web Start. When you undeploy an application client, you can no longer use Java Web Start, or any other mechanism, to download the application client. 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 depend on whether the application client has been written to tolerate such situations.

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

Deploying a J2EE CA Connector Module

You deploy a J2EE Connector Architecture (CA) connector module as described in "Tools for Deployment" on page 37. After deploying the module, you must configure it as described in Chapter 12, "Developing Connectors," in *Sun Java System Application Server 9.1 Developer's Guide*.

Access to Shared Frameworks

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 System Classloader, the Common Classloader, or an application-specific class loader rather than in an application or module. 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 classloaders, which is a waste of resources. For more information, see Chapter 2, "Class Loaders," in *Sun Java System Application Server 9.1 Developer's Guide*.

◆ ◆ ◆ APPENDIX A

Deployment Descriptor Files

This chapter describes deployment descriptor files specific to the Sun Java System Application Server in the following sections:

Sun Java System Application Server Descriptors

Sun Java System Application Server uses optional deployment descriptors in addition to the Java EE standard descriptors for configuring features specific to the Application Server.

Note – Settings in the Application Server deployment descriptors override corresponding settings in the Java EE deployment descriptors and in the Application Server's `domain.xml` file unless otherwise stated. For more information about the `domain.xml` file, see the *Sun Java System Application Server 9.1 Administration Reference*.

Each deployment descriptor (or XML) file has a corresponding DTD file, which defines the elements, data, and attributes that the deployment descriptor file can contain. For example, the `sun-application_5_0-0.dtd` file defines the structure of the `sun-application.xml` file. The DTD files for the Application Server deployment descriptors are located in the *as-install*`/lib/dtds` directory.

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

To check the correctness of these deployment descriptors prior to deployment, see "The `verifier` Utility" on page 26.

For general information about DTD files and XML, see the XML specification at `http://www.w3.org/TR/REC-xml`.

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

TABLE A–1 Sun Java System Application Server Descriptors

Deployment Descriptor	DTD File	Description
`sun-application.xml`	`sun-application_5_0-0.dtd`	Configures an entire Java EE application (EAR file).
`sun-web.xml`	`sun-web-app_2_5-0.dtd`	Configures a web application (WAR file).
`sun-ejb-jar.xml`	`sun-ejb-jar_3_0-0.dtd`	Configures an enterprise bean (EJB JAR file).
`sun-cmp-mappings.xml`	`sun-cmp-mapping_1_2.dtd`	Configures container-managed persistence for an enterprise bean.
`sun-application-client.xml`	`sun-application-client_5_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. The Application Server provides a default file in the *domain-dir*`/config` directory. Specifying a different file is optional.

Note – The Application Server deployment descriptors 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.

The sun-application.xml File

The element hierarchy in the `sun-application.xml` file 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
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-application PUBLIC '-//Sun Microsystems, Inc.//DTD
Application Server 9.0 Java EE Application 5.0//EN'
'http://www.sun.com/software/appserver/dtds/sun-application_5_0-0.dtd'>
<sun-application>
   <unique-id>67488732739338240</unique-id>
</sun-application>
```

The sun-web.xml File

The element hierarchy in the `sun-web.xml` file 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:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC '-//Sun Microsystems, Inc.//DTD
Application Server 9.0 Servlet 2.5//EN'
'http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.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 element hierarchy in the `sun-ejb-jar.xml` file 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:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD
Application Server 9.0 EJB 3.0//EN'
'http://www.sun.com/software/appserver/dtds/sun-ejb-jar_3_0-0.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 element hierarchy in the `sun-cmp-mappings.xml` file 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 element hierarchy in the `sun-application-client.xml` file 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
Application Server 9.0 Application Client 5.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-application-client_5_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 element hierarchy in the `sun-acc.xml` file 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)
```

Alphabetical Listing of All Elements

"A" on page 63 "B" on page 67 "C" on page 69 "D" on page 89 "E" on page 92 "F" on page 100 "G" on page 102 "H" on page 103 "I" on page 104 "J" on page 106 "K" on page 112 "L" on page 112 "M" on page 117 "N" on page 128 "O" on page 129 "P" on page 130 "Q" on page 138 "R" on page 139 "S" on page 148 "T" on page 168 "U" on page 172 "V" on page 173 "W" on page 175

A

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 121 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–2 `activation-config` subelements

Element	Required	Description
"description" on page 91	zero or one	Specifies a text description of the activation configuration.
"activation-config-property" on page 64	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 63 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–3 `activation-config-property` subelements

Element	Required	Description
"activation-config-property-name" on page 64	only one	Specifies the name of an activation configuration property.
"activation-config-property-value" on page 65	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 64 (`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 64 (`sun-ejb-jar.xml`)

Subelements

none - contains data

as-context

Specifies the authentication mechanism used to authenticate the client.

Superelements

"ior-security-config" on page 105 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–4 `as-context` Subelements

Element	Required	Description
"auth-method" on page 65	only one	Specifies the authentication method. The only supported value is `USERNAME_PASSWORD`.
"realm" on page 139	only one	Specifies the realm in which the user is authenticated.
"required" on page 143	only one	Specifies whether the authentication method specified in the `auth-method` element must be used for client authentication.

auth-method

Specifies the authentication method.

If the parent element is "as-context" on page 65, the only supported value is `USERNAME_PASSWORD`.

If the parent element is "login-config" on page 117, 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 117 (`sun-web.xml`), "as-context" on page 65 (`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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"client-container" on page 77 (`sun-acc.xml`)

Subelements

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

TABLE A–5 `auth-realm` subelement

Element	Required	Description
"property (with attributes)" on page 135	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 A–6 `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 *Sun Java System Application Server 9.1 Developer's Guide*.

B

bean-cache

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

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–7 `bean-cache` Subelements

Element	Required	Description
"max-cache-size" on page 119	zero or one	Specifies the maximum number of beans allowable in cache.
"is-cache-overflow-allowed" on page 105	zero or one	Deprecated.
"cache-idle-timeout-in-seconds" on page 71	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 141	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 144	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 174	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 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–8 `bean-pool` Subelements

Element	Required	Description
"steady-pool-size" on page 158	zero or one	Specifies the initial and minimum number of beans maintained in the pool. Default is 32.
"resize-quantity" on page 144	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 120	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 120	zero or one	Deprecated.
"pool-idle-timeout-in-seconds" on page 132	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>
```

C

cache

Configures caching for web application components.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

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

TABLE A–9 `cache` Subelements

Element	Required	Description
"cache-helper" on page 70	zero or more	Specifies a custom class that implements the `CacheHelper` interface.
"default-helper" on page 90	zero or one	Allows you to change the properties of the default, built-in "cache-helper" on page 70 class.
"property (with attributes)" on page 135	zero or more	Specifies a cache property, which has a name and a value.
"cache-mapping" on page 72	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 A–10 `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 170 element.

TABLE A–10 cache Attributes *(Continued)*

Attribute	Default	Description
enabled	true	(optional) Determines whether servlet and JSP caching is enabled.

Properties

The following table describes properties for the cache element.

TABLE A–11 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 70 for possible values.
MultiLRUSegmentSize	4096	Specifies the number of entries in a segment of the cache table that should have its own LRU (least recently used) list. Applicable only if cacheClassName is set to com.sun.appserv.web.cache.MultiLruCache.
MaxSize	unlimited; Long.MAX_VALUE	Specifies an upper bound on the cache memory size in bytes (KB or MB units). Example values are 32 KB or 2 MB. Applicable only if cacheClassName is set to com.sun.appserv.web.cache.BoundedMultiLruCache.

Cache Class Names

The following table lists possible values of the cacheClassName property.

TABLE A–12 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 69 (`sun-web.xml`)

Subelements

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

TABLE A–13 `cache-helper` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–14 `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 72 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 70 used by the parent "cache-mapping" on page 72 element.

Superelements

"cache-mapping" on page 72 (`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 67 (`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 69 (`sun-web.xml`)

Subelements

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

TABLE A–15 `cache-mapping` Subelements

Element	Required	Description
"servlet-name" on page 155	requires one `servlet-name` or `url-pattern`	Contains the name of a servlet.
"url-pattern" on page 172	requires one `servlet-name` or `url-pattern`	Contains a servlet URL pattern for which caching is enabled.
"cache-helper-ref" on page 71	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 70 used by the parent `cache-mapping` element.
"dispatcher" on page 91	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 170	zero or one if `cache-helper-ref` is not used	Contains the "cache-mapping" on page 72 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 140	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 103	zero or more if `cache-helper-ref` is not used	Contains an HTTP method that is eligible for caching.

TABLE A–15 `cache-mapping` Subelements *(Continued)*

Element	Required	Description
"key-field" on page 112	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 85	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 133, "service-ref" on page 153 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–16 `call-property` subelements

Element	Required	Description
"name" on page 128	only one	Specifies the name of the entity.
"value" on page 173	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 148 (`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 151 (`sun-acc.xml`)

Subelements

none

Attributes

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

TABLE A–17 `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 84 (`sun-cmp-mappings.xml`)

check-modified-at-commit

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

Superelements

"consistency" on page 84 (`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 84 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–18 `check-version-of-accessed-instances` Subelements

Element	Required	Description
"column-name" on page 83	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 92 element must be set to `true`.

Note – Some topics in the documentation pertain to features that are available only in domains that are configured to support clusters. Examples of domains that support clusters are domains that are created with the cluster profile or the enterprise profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*.

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–19 `checkpoint-at-end-of-method` Subelements

Element	Required	Description
"method" on page 126	one or more	Specifies a bean method.

checkpointed-methods

Deprecated. Supported for backward compatibility. Use "checkpoint-at-end-of-method" on page 75 instead.

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

class-loader

Configures the class loader for the web module.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

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

TABLE A–20 `class-loader` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–21 `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"`

TABLE A–21 `class-loader` Attributes *(Continued)*

Attribute	Default	Description
`delegate`	`true`	(optional) If `true`, the web module follows the standard class loader delegation model and delegates to its parent class loader first before looking in the local class loader. You must set this to `true` for a web application 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.
`dynamic-reload-interval`		(optional) Not implemented. Included for backward compatibility with previous Sun Java System Web Server versions.

Note – If the `delegate` element 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 A–22 `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 Application 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 63.

Superelements

none

Subelements

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

TABLE A–23 `client-container` Subelements

Element	Required	Description
"target-server" on page 169	only one (developer profile) one or more (cluster and enterprise profiles)	Specifies the IIOP listener for the target server. Also specifies IIOP endpoints used for load balancing. If the Application Server instance on which the application client is deployed participates in a cluster, the Application Server finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed. **Note –** Some topics in the documentation pertain to features that are available only in domains that are configured to support clusters. Examples of domains that support clusters are domains that are created with the cluster profile or the enterprise profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*. 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 66	zero or one	Specifies the optional configuration for JAAS authentication realm.
"client-credential" on page 79	zero or one	Specifies the default client credential that is sent to the server.
"log-service" on page 116	zero or one	Specifies the default log file and the severity level of the message.
"message-security-config" on page 125	zero or more	Specifies configurations for message security providers.
"property (with attributes)" on page 135	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 A–24 `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 A–25 `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 169 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 77 (`sun-acc.xml`)

Subelements

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

TABLE A–26 `client-credential` subelement

Element	Required	Description
"property (with attributes)" on page 135	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 A–27 `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 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–28 `cmp` Subelements

Element	Required	Description
"mapping-properties" on page 119	zero or one	This element is not implemented.
"is-one-one-cmp" on page 105	zero or one	This element is not implemented.
"one-one-finders" on page 129	zero or one	Describes the finders for CMP 1.1 beans.
"prefetch-disabled" on page 133	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 99 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–29 `cmp-field-mapping` Subelements

Element	Required	Description
"field-name" on page 101	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 83	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.

TABLE A–29 `cmp-field-mapping` Subelements *(Continued)*

Element	Required	Description
"read-only" on page 139	zero or one	Specifies that a field is read-only.
"fetched-with" on page 100	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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"enterprise-beans" on page 97 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–30 `cmp-resource` Subelements

Element	Required	Description
"jndi-name" on page 108	only one	Specifies the absolute `jndi-name` of a JDBC resource.
"default-resource-principal" on page 90	zero or one	Specifies the default runtime bindings of a resource reference.
"property (with subelements)" on page 136	zero or more	Specifies a property name and value. Used to configure `PersistenceManagerFactory` properties.
"create-tables-at-deploy" on page 88	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 92	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 89	zero or one	Specifies the name of the database vendor for which tables can be created.
"schema-generator-properties" on page 149	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 99 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–31 `cmr-field-mapping` Subelements

Element	Required	Description
"cmr-field-name" on page 82	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 83	one or more	Specifies the pair of columns that determine the relationship between two database tables.
"fetched-with" on page 100	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 82 (`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 92 (`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 75, "cmp-field-mapping" on page 80, "column-pair" on page 83 (`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 82, "secondary-table" on page 151 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–32 `column-pair` Subelements

Element	Required	Description
"column-name" on page 83	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 the Application Server are `B` or `C`. Default value is `B`. Applies to entity beans.

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

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

none - contains data

confidentiality

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

Superelements

"transport-config" on page 171 (`sun-ejb-jar.xml`)

Subelements

none - contains data

consistency

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

Superelements

"entity-mapping" on page 99 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–33 `consistency` Subelements

Element	Required	Description
"none" on page 129	exactly one subelement is required	No consistency checking occurs.
"check-modified-at-commit" on page 74	exactly one subelement is required	Checks concurrent modification of fields in modified beans at commit time.
"lock-when-loaded" on page 116	exactly one subelement is required	Obtains an exclusive lock when the data is loaded.
"check-all-at-commit" on page 74		This element is not implemented. Do not use.
"lock-when-modified" on page 116		This element is not implemented. Do not use.
"check-version-of-accessed-instances" on page 75	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 172 or "servlet-name" on page 155.

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 72 (`sun-web.xml`)

Subelements

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

TABLE A–34 `constraint-field` Subelements

Element	Required	Description
"constraint-field-value" on page 86	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 A–35 `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 86 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 86 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 85 (`sun-web.xml`)

Subelements

none - contains data

Attributes

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

TABLE A–36 `constraint-field-value` Attributes

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

TABLE A–36 `constraint-field-value` Attributes *(Continued)*

Attribute	Default	Description
`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. 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 106.

If you are setting up load balancing, web module context roots must be unique within a server instance. See the *Sun Java System Application Server 9.1 High Availability Administration Guide* for more information about load balancing.

Note – Some topics in the documentation pertain to features that are available only in domains that are configured to support clusters. Examples of domains that support clusters are domains that are created with the cluster profile or the enterprise profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*.

Superelements

"web" on page 175 (`sun-application.xml`), "sun-web-app" on page 164 (`sun-web.xml`), "java-web-start-access" on page 106 (`sun-application-client.xml`)

Subelements

none - contains data

cookie-properties

Specifies session cookie properties.

Superelements

"session-config" on page 155 (`sun-web.xml`)

Subelements

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

TABLE A–37 `cookie-properties` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–38 `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 the Application 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 Java System Application 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.

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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"cmp-resource" on page 81 (`sun-ejb-jar.xml`)

Subelements

none - contains data

D

database-vendor-name

Specifies the name of the database vendor for which tables can be created. Allowed values are `javadb`, `db2`, `mssql`, `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 108 subelement of the "cmp-resource" on page 81 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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"cmp-resource" on page 81 (`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 176 (`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 133 element in the `sun-ejb-jar.xml` file.

Superelements

"fetched-with" on page 100 (`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 70 class.

Superelements

"cache" on page 69 (`sun-web.xml`)

Subelements

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

TABLE A–39 `default-helper` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–40 `default-helper` Properties

Property	Default	Description
`cacheKeyGeneratorAttrName`	Uses the built-in `default` "cache-helper" on page 70 key generation, which concatenates the servlet path with "key-field" on page 112 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 *Sun Java System Application Server 9.1 Developer's 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 Sun Java™ System Message Queue broker user repository. See the *Security Management* chapter in the *Sun Java System Message Queue 4.1 Administration Guide* for details.

Superelements

"resource-ref" on page 146 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "cmp-resource" on page 81, "mdb-connection-factory" on page 120 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–41 `default-resource-principal` Subelements

Element	Required	Description
"name" on page 128	only one	Specifies the default resource principal name used to sign on to a resource manager.
"password" on page 131	only one	Specifies password of the default resource principal.

description

Specifies a text description of the containing element.

Superelements

"property (with attributes)" on page 135 (`sun-web.xml`); "activation-config" on page 63, "method" on page 126 (`sun-ejb-jar.xml`); "target-server" on page 169 (`sun-acc.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 72 (`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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"cmp-resource" on page 81 (`sun-ejb-jar.xml`)

Subelements

none - contains data

E

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 97 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–42 `ejb` Subelements

Element	Required	Description
"ejb-name" on page 95	only one	Matches the `ejb-name` in the corresponding `ejb-jar.xml` file.
"jndi-name" on page 108	zero or more	Specifies the absolute `jndi-name`.
"ejb-ref" on page 95	zero or more	Maps the absolute JNDI name to the `ejb-ref` element in the corresponding Java EE XML file.
"resource-ref" on page 146	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Java EE XML file.
"resource-env-ref" on page 145	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Java EE XML file.
"service-ref" on page 153	zero or more	Specifies runtime settings for a web service reference.
"message-destination-ref" on page 123	zero or more	Specifies the name of a physical message destination.
"pass-by-reference" on page 130	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 79	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 134	zero or one	Specifies the principal (user) name in an enterprise bean that has the `run-as` role specified.
"mdb-connection-factory" on page 120	zero or one	Specifies the connection factory associated with a message-driven bean.
"jms-durable-subscription-name" on page 107	zero or one	Specifies the durable subscription associated with a message-driven bean.
"jms-max-messages-load" on page 107	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 105	zero or one	Specifies the security information for the IOR.
"is-read-only-bean" on page 105	zero or one	Specifies that this entity bean is read-only.
"refresh-period-in-seconds" on page 140	zero or one	Specifies the rate at which a read-only-bean must be refreshed from the data source.
"commit-option" on page 84	zero or one	Has valid values of B or C. Default value is B.

TABLE A–42 `ejb` Subelements *(Continued)*

Element	Required	Description
"cmt-timeout-in-seconds" on page 82	zero or one	Overrides the Transaction Timeout setting of the Transaction Service for an individual bean.
"use-thread-pool-id" on page 173	zero or one	Specifies the thread pool from which threads are selected for remote invocations of this bean.
"gen-classes" on page 102	zero or one	Specifies all the generated class names for a bean.
"bean-pool" on page 68	zero or one	Specifies the bean pool properties. Used for stateless session beans, entity beans, and message-driven beans.
"bean-cache" on page 67	zero or one	Specifies the bean cache properties. Used only for stateful session beans and entity beans.
"mdb-resource-adapter" on page 121	zero or one	Specifies runtime configuration information for a message-driven bean.
"webservice-endpoint" on page 176	zero or more	Specifies information about a web service endpoint.
"flush-at-end-of-method" on page 102	zero or one	Specifies the methods that force a database flush after execution. Used for entity beans.
"checkpointed-methods" on page 76	zero or one	Deprecated. Supported for backward compatibility. Use "checkpoint-at-end-of-method" on page 75 instead.
"checkpoint-at-end-of-method" on page 75	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 A–43 `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 92, "method" on page 126 (`sun-ejb-jar.xml`); "entity-mapping" on page 99 (`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 164 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`), "sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–44 `ejb-ref` Subelements

Element	Required	Description
"ejb-ref-name" on page 96	only one	Specifies the `ejb-ref-name` in the corresponding Java EE deployment descriptor file `ejb-ref` entry.
"jndi-name" on page 108	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 95 (`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 106 (`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 176 (`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 163 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–45 `enterprise-beans` Subelements

Element	Required	Description
"name" on page 128	zero or one	Specifies the name string.
"unique-id" on page 172	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 92	zero or more	Defines runtime properties for a single enterprise bean within the application.
"pm-descriptors" on page 132	zero or one	Deprecated.
"cmp-resource" on page 81	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 122	zero or more	Specifies the name of a logical message destination.
"webservice-description" on page 175	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 162 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–46 `entity-mapping` Subelements

Element	Required	Description
"ejb-name" on page 95	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 168	only one	Specifies the name of a database table. The table must be present in the database schema file.
"cmp-field-mapping" on page 80	one or more	Associates a field with one or more columns to which it maps.
"cmr-field-mapping" on page 82	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 151	zero or more	Describes the relationship between a bean's primary and secondary table.
"consistency" on page 84	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 171 (`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 171 (`sun-ejb-jar.xml`)

Subelements

none - contains data

F

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 80, 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 80, 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 82, `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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"cmp-field-mapping" on page 80, "cmr-field-mapping" on page 82 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–47 `fetched-with` Subelements

Element	Required	Description
"default" on page 89	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 112	exactly one subelement is required	Specifies the level number of a hierarchical fetch group.
"named-group" on page 128	exactly one subelement is required	Specifies the name of an independent fetch group.
"none" on page 129	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 80 (`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 129 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–48 `finder` Subelements

Element	Required	Description
"method-name" on page 127	only one	Specifies the method name for the finder.
"query-params" on page 139	zero or one	Specifies the query parameters for the CMP 1.1 finder.
"query-filter" on page 138	zero or one	Specifies the query filter for the CMP 1.1 finder.
"query-variables" on page 139	zero or one	Specifies variables in query expression for the CMP 1.1 finder.
"query-ordering" on page 138	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 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–49 `flush-at-end-of-method` Subelements

Element	Required	Description
"method" on page 126	one or more	Specifies a bean method.

G

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 92 (`sun-ejb-jar.xml`)

Subelements

The following table describes subelements for the `gen-class` element.

TABLE A–50 `gen-classes` Subelements

Element	Required	Description
"remote-impl" on page 141	zero or one	Specifies the fully-qualified class name of the generated `EJBObject` impl class.
"local-impl" on page 113	zero or one	Specifies the fully-qualified class name of the generated `EJBLocalObject` impl class.
"remote-home-impl" on page 141	zero or one	Specifies the fully-qualified class name of the generated EJBHome impl class.
"local-home-impl" on page 113	zero or one	Specifies the fully-qualified class name of the generated `EJBLocalHome` impl class.

group-name

Specifies a group name in the current realm.

Superelements

"security-role-mapping" on page 151 (`sun-application.xml`, `sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

H

http-method

Specifies an HTTP method that is eligible for caching. The default is `GET`.

Superelements

"cache-mapping" on page 72 (`sun-web.xml`)

Subelements

none - contains data

I

idempotent-url-pattern

Specifies a URL pattern for idempotent requests.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

none

Attributes

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

TABLE A–51 `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 171 (`sun-ejb-jar.xml`)

Subelements

none - contains data

ior-security-config

Specifies the security information for the input-output redirection (IOR).

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–52 `ior-security-config` Subelements

Element	Required	Description
"transport-config" on page 171	zero or one	Specifies the security information for transport.
"as-context" on page 65	zero or one	Specifies the authentication mechanism used to authenticate the client. If specified, it is `USERNAME_PASSWORD`.
"sas-context" on page 148	zero or one	Describes the sas-context fields.

is-cache-overflow-allowed

This element is deprecated. Do not use.

Superelements

"bean-cache" on page 67 (`sun-ejb-jar.xml`)

is-one-one-cmp

This element is not used.

Superelements

"cmp" on page 79 (`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.

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

none - contains data

J

java-method

Specifies a method.

Superelements

"message" on page 121 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–53 `java-method` Subelements

Element	Required	Description
"method-name" on page 127	only one	Specifies a method name.
"method-params" on page 127	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 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–54 `java-web-start-access` subelements

Element	Required	Description
"context-root" on page 87	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 an application 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 `app-name` or `module-name` is not specified during deployment, the name of the EAR or JAR file without the extension is used. If the application or module is not in EAR or JAR file format, a name is generated and written to the server log.
"eligible" on page 96	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 173	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`.

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 92 (`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 92 (`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 63 subelement of the "mdb-resource-adapter" on page 121 element. For more information about JMS resources, see Chapter 18, "Using the Java Message Service," in *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"ejb-ref" on page 95, "message-destination" on page 122, "resource-env-ref" on page 145, "resource-ref" on page 146 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "cmp-resource" on page 81, "ejb" on page 92, "mdb-connection-factory" on page 120 (`sun-ejb-jar.xml`)

Subelements

none - contains data

jsp-config

Specifies JSP configuration information.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

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

TABLE A–55 `jsp-config` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–56 `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 the Application 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/j2se/1.5.0/docs/tooldocs/solaris/javac.html#options.

TABLE A–56 `jsp-config` Properties *(Continued)*

Property	Default	Description
`compilerTargetVM`	Depends on the Application Server's Java runtime	Specifies the JVM 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/j2se/1.5.0/docs/tooldocs/solaris/javac.html#options.
`defaultBufferNone`	`false`	If `true`, the default for the `buffer` attribute of the `page` directive is `none`.
`development`	`true`	If set to `true`, enables development mode, which allows JSP files to be checked for modification. Specify the frequency at which JSPs are checked using the `modificationTestInterval` property.
`dumpSmap`	`false`	If set to `true`, dumps SMAP information for JSR 45 debugging to a file. Set to `false` if `suppressSmap` is `true`.
`enablePooling`	`true`	If set to `true`, tag handler pooling is enabled.
`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 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 A–56 `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/j2se/1.5.0/docs/guide/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.

K

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 72 (`sun-web.xml`)

Subelements

none

Attributes

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

TABLE A–57 `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`.

L

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 100 (`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 102 (`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 102 (`sun-ejb-jar.xml`)

Subelements

none - contains data

locale-charset-info

Deprecated. For backward compatibility only. Use the "parameter-encoding" on page 130 subelement of "sun-web-app" on page 164 instead. Specifies information about the application's internationalization settings.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

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

TABLE A–58 `locale-charset-info` Subelements

Element	Required	Description
"locale-charset-map" on page 114	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 130	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 A–59 `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 130 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 130 element. For encodings, see `http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html`.

Superelements

"locale-charset-info" on page 113 (`sun-web.xml`)

Subelements

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

TABLE A–60 `locale-charset-map` Subelements

Element	Required	Description
"description" on page 91	zero or one	Specifies an optional text description of a mapping.

Attributes

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

TABLE A–61 `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 application 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 A–62 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 A–62 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 153, "wsdl-port" on page 178 (`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 84 (`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 84 (`sun-cmp-mappings.xml`)

log-service

Specifies configuration settings for the log file.

Superelements

"client-container" on page 77 (`sun-acc.xml`)

Subelements

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

TABLE A–63 `log-service` subelement

Element	Required	Description
"property (with attributes)" on page 135	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 A–64 `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 176 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

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

TABLE A–65 `login-config` subelements

Element	Required	Description
"auth-method" on page 65	only one	Specifies the authentication method.
"realm" on page 139	zero or one	Specifies the name of the realm used to process all authentication requests.

M

manager-properties

Specifies session manager properties.

Superelements

"session-manager" on page 156 (`sun-web.xml`)

Subelements

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

TABLE A–66 `manager-properties` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–67 `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 156 element is `file`, `ha`, or `replicated`.
`maxSessions`	`-1`	Specifies the maximum number of sessions that are permitted in the cache, or `-1` for no limit. After this, an attempt to create a new session causes an `IllegalStateException` to be thrown. If the `persistence-type` attribute of the parent "session-manager" on page 156 element is `file` or `replicated`, the session manager passivates sessions to the persistent store when this maximum is reached.
`sessionFilename`	One of the following: *domain-dir*`/generated/jsp/j2ee-modules/`*module-name*`/`*context-path*`_SESSIONS.ser` *domain-dir*`/generated/jsp/j2ee-apps/`*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 application. 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 156 element is `memory`.

TABLE A–67 `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 156 element is `ha` or `replicated`.

mapping-properties

This element is not implemented.

Superelements

"cmp" on page 79 (`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 67 (`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 68 (`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 68 (`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 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–68 `mdb-connection-factory` Subelements

Element	Required	Description
"jndi-name" on page 108	only one	Specifies the absolute `jndi-name`.

TABLE A–68 `mdb-connection-factory` Subelements *(Continued)*

Element	Required	Description
"default-resource-principal" on page 90	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 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–69 `mdb-resource-adapter` subelements

Element	Required	Description
"resource-adapter-mid" on page 144	zero or one	Specifies a resource adapter module ID.
"activation-config" on page 63	one or more	Specifies an activation configuration.

message

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

Superelements

"message-security" on page 123 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–70 `message` Subelements

Element	Required	Description
"java-method" on page 106	zero or one	Specifies the methods or operations to which message security requirements apply.
"operation-name" on page 129	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 164 (`sun-web.xml`), "enterprise-beans" on page 97 (`sun-ejb-jar.xml`), "sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–71 `message-destination` subelements

Element	Required	Description
"message-destination-name" on page 122	only one	Specifies the name of a logical message destination defined within the corresponding Java EE deployment descriptor file.
"jndi-name" on page 108	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 122 (`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 164 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`), "sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–72 `message-destination-ref` subelements

Element	Required	Description
"message-destination-ref-name" on page 123	only one	Specifies the name of a physical message destination defined within the corresponding Java EE deployment descriptor file.
"jndi-name" on page 108	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 123 (`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 176, these requirements pertain to request and response messages of the endpoint.
- If the grandparent element is "port-info" on page 133, these requirements pertain to the port of the referenced service.

Superelements

"message-security-binding" on page 124 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–73 `message-security` Subelements

Element	Required	Description
"message" on page 121	one or more	Specifies the methods or operations to which message security requirements apply.
"request-protection" on page 142	zero or one	Defines the authentication policy requirements of the application's request processing.
"response-protection" on page 147	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 176 or "port-info" on page 133 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 176, "port-info" on page 133 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–74 `message-security-binding` Subelements

Element	Required	Description
"message-security" on page 123	zero or more	Specifies message security requirements.

Attributes

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

TABLE A–75 `message-security-binding` Attributes

Attribute	Default	Description
`auth-layer`	none	Specifies the message layer at which authentication is performed. The value must be `SOAP`.
`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 77 (`sun-acc.xml`)

Subelements

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

TABLE A–76 `message-security-config` Subelements

Element	Required	Description
"provider-config" on page 136	one or more	Specifies a configuration for one message security provider.

Attributes

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

TABLE A–77 `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

"flush-at-end-of-method" on page 102 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–78 `method` Subelements

Element	Required	Description
"description" on page 91	zero or one	Specifies an optional text description.
"ejb-name" on page 95	zero or one	Matches the `ejb-name` in the corresponding `ejb-jar.xml` file.
"method-name" on page 127	only one	Specifies a method name.
"method-intf" on page 126	zero or one	Specifies the method interface to distinguish between methods with the same name in different interfaces.
"method-params" on page 127	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 126 (`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 106 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "finder" on page 101, "query-method" on page 138 , "method" on page 126 (`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 127 (`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 106 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "query-method" on page 138, "method" on page 126 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–79 `method-params` Subelements

Element	Required	Description
"method-param" on page 127	zero or more	Specifies the fully qualified Java type name of a method parameter.

N

name

Specifies the name of the entity.

Superelements

"call-property" on page 73, "default-resource-principal" on page 90, "stub-property" on page 160 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "enterprise-beans" on page 97, "principal" on page 134, "property (with subelements)" on page 136 (`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 100 (`sun-cmp-mappings.xml`)

Subelements

none - contains data

namespaceURI

Specifies the namespace URI.

Superelements

"service-qname" on page 153, "wsdl-port" on page 178 (`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 84, "fetched-with" on page 100 (`sun-cmp-mappings.xml`)

Subelements

none - element is present or absent

O

one-one-finders

Describes the finders for CMP 1.1 beans.

Superelements

"cmp" on page 79 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–80 `one-one-finders` Subelements

Element	Required	Description
"finder" on page 101	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 121 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

none - contains data

P

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 164 and "locale-charset-info" on page 113 elements have `parameter-encoding` subelements, the subelement of `sun-web-app` takes precedence. For encodings, see `http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html`.

Superelements

"locale-charset-info" on page 113, "sun-web-app" on page 164 (`sun-web.xml`)

Subelements

none

Attributes

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

TABLE A–81 `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 the Application 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 161 (`sun-application.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

none - contains data

password

Specifies the password for the principal.

Superelements

"default-resource-principal" on page 90 (`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 97 (`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 68 (`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 176 (`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 153 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–82 `port-info` subelements

Element	Required	Description
"service-endpoint-interface" on page 152	zero or one	Specifies the web service reference name relative to `java:comp/env`.
"wsdl-port" on page 178	zero or one	Specifies the WSDL port.
"stub-property" on page 160	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 73	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 124	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 100 element is set to "default" on page 89.

Superelements

"cmp" on page 79 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–83 `prefetch-disabled` Subelements

Element	Required	Description
"query-method" on page 138	one or more	Specifies a query method.

principal

Defines a node that specifies a user name on the platform.

Superelements

"ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–84 `principal` Subelements

Element	Required	Description
"name" on page 128	only one	Specifies the name of the user.

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 151 (`sun-application.xml`, `sun-web.xml`, `sun-ejb-jar.xml`), "servlet" on page 154 (`sun-web.xml`)

Subelements

none - contains data

Attributes

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

TABLE A–85 `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 Application Server
- Needed by a system or object that Application Server doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

"cache" on page 69, "cache-helper" on page 70, "class-loader" on page 76, "cookie-properties" on page 87, "default-helper" on page 90, "manager-properties" on page 117, "session-properties" on page 156, "store-properties" on page 158, "sun-web-app" on page 164, "webservice-endpoint" on page 176 (`sun-web.xml`); "auth-realm" on page 66, "client-container" on page 77, "client-credential" on page 79, "log-service" on page 116, "provider-config" on page 136 (`sun-acc.xml`)

Subelements

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

TABLE A–86 `property` Subelements

Element	Required	Description
"description" on page 91	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 A–87 `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 Application Server
- Needed by a system or object that Application Server doesn't have knowledge of, such as an LDAP server or a Java class

Superelements

"cmp-resource" on page 81, "schema-generator-properties" on page 149, "webservice-endpoint" on page 176 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–88 `property` subelements

Element	Required	Description
"name" on page 128	only one	Specifies the name of the property.
"value" on page 173	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 125 (`sun-acc.xml`)

Subelements

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

TABLE A–89 `provider-config` Subelements

Element	Required	Description
"request-policy" on page 142	zero or one	Defines the authentication policy requirements of the authentication provider's request processing.
"response-policy" on page 146	zero or one	Defines the authentication policy requirements of the authentication provider's response processing.
"property (with attributes)" on page 135	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 A–90 `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.

Q

query-filter

Specifies the query filter for the CMP 1.1 finder.

Superelements

"finder" on page 101 (`sun-ejb-jar.xml`)

Subelements

none - contains data

query-method

Specifies a query method.

Superelements

"prefetch-disabled" on page 133 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–91 `query-method` Subelements

Element	Required	Description
"method-name" on page 127	only one	Specifies a method name.
"method-params" on page 127	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 101 (`sun-ejb-jar.xml`)

Subelements

none - contains data

query-params

Specifies the query parameters for the CMP 1.1 finder.

Superelements

"finder" on page 101 (`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 101 (`sun-ejb-jar.xml`)

Subelements

none - contains data

R

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 80 (`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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"sun-application" on page 161 (`sun-application.xml`), "as-context" on page 65, "login-config" on page 117 (`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 72 (`sun-web.xml`)

Subelements

none

Attributes

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

TABLE A–92 `refresh-field` Attributes

Attribute	Default	Description
`name`	none	Specifies the input parameter name.
`scope`	`request.parameter`	(optional) Specifies the scope from which the input parameter is retrieved. Allowed values are `context.attribute`, `request.header`, `request.parameter`, `request.cookie`, `session.id`, and `session.attribute`.

refresh-period-in-seconds

Specifies the rate at which a read-only-bean must be refreshed from the data source. If the value is less than or equal to zero, the bean is never refreshed; if the value is greater than zero, the bean instances are refreshed at the specified interval. This rate is just a hint to the container. Default is 0 (no refresh).

Superelements

"ejb" on page 92 (`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 71.

Superelements

"bean-cache" on page 67 (`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 102 (`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 102 (`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 136 (`sun-acc.xml`)

Subelements

none

Attributes

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

TABLE A–93 `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 123 (`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 A–94 `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 65 element must be used for client authentication. The value is `true` or `false` (the default).

Superelements

"as-context" on page 65 (`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 146 (`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 158 quantity of beans (applies to pools only for creation). If the pool has more than `steady-pool-size` minus "resize-quantity" on page 144 of beans, then `resize-quantity` is still created.
- Removed, when the "pool-idle-timeout-in-seconds" on page 132 timer expires and a cleaner thread removes any unused instances.
  - For caches, when "max-cache-size" on page 119 is reached, `resize-quantity` beans are selected for passivation using the "victim-selection-policy" on page 174. In addition, the "cache-idle-timeout-in-seconds" on page 71 or "removal-timeout-in-seconds" on page 141 timers passivate beans from the cache.
  - For pools, when the "max-pool-size" on page 120 is reached, `resize-quantity` beans are selected for removal. In addition, the "pool-idle-timeout-in-seconds" on page 132 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 67, "bean-pool" on page 68 (`sun-ejb-jar.xml`)

Subelements

none - contains data

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 121 (`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 164 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`), "sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–95 `resource-env-ref` Subelements

Element	Required	Description
"resource-env-ref-name" on page 145	only one	Specifies the `res-ref-name` in the corresponding Java EE deployment descriptor file `resource-env-ref` entry.
"jndi-name" on page 108	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 145 (`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 the Application 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 Sun Java System Message Queue, the `default-resource-principal` (name/password) must exist in the Message Queue user repository. Refer to the *Security Management* chapter in the *Sun Java System Message Queue 4.1 Administration Guide* for information on how to manage the Message Queue user repository.

Superelements

"sun-web-app" on page 164 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`), "sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–96 `resource-ref` Subelements

Element	Required	Description
"res-ref-name" on page 143	only one	Specifies the `res-ref-name` in the corresponding Java EE deployment descriptor file `resource-ref` entry.
"jndi-name" on page 108	only one	Specifies the absolute `jndi-name` of a resource.
"default-resource-principal" on page 90	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>
```

response-policy

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

Superelements

"provider-config" on page 136 (`sun-acc.xml`)

Subelements

none

Attributes

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

TABLE A–97 `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 123 (`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 A–98 `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).

TABLE A–98 `response-protection` Attributes *(Continued)*

Attribute	Default	Description
`auth-recipient`	none	Specifies whether recipient authentication occurs before or after content authentication. Allowed values are `before-content` and `after-content`.

role-name

Contains the `role-name` in the `security-role` element of the corresponding Java EE deployment descriptor file.

Superelements

"security-role-mapping" on page 151 (`sun-application.xml`, `sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

S

sas-context

Describes the sas-context fields.

Superelements

"ior-security-config" on page 105 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–99 `sas-context` Subelements

Element	Required	Description
"caller-propagation" on page 73	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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"sun-cmp-mapping" on page 162 (`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 81 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–100 `schema-generator-properties` Subelements

Element	Required	Description
"property (with subelements)" on page 136	zero or more	Specifies a property name and value.

Properties

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

TABLE A–101 `schema-generator-properties` Properties

Property	Default	Description
`use-unique-table-names`	`false`	Specifies that generated table names are unique within each application server domain. This property can be overridden during deployment. See "Generation Options for CMP" in *Sun Java System Application Server 9.1 Developer's Guide*.
bean-name.*field-name*.*attribute*	none	Defines a column attribute. For attribute descriptions, see Table A–102.

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

TABLE A–102 `schema-generator-properties` Column Attributes

Attribute	Description
`jdbc-type`	Specifies the JDBC type of the column created for the CMP field. The actual SQL type generated is based on this JDBC type but is database vendor specific.
`jdbc-maximum-length`	Specifies the maximum number of characters stored in the column corresponding to the CMP field. Applies only when the actual SQL that is generated for the column requires a length. For example, a `jdbc-maximum-length` of `32` on a CMP `String` field such as `firstName` normally results in a column definition such as VARCHAR(32). But if the `jdbc-type` is `CLOB` and you are deploying on Oracle, the resulting column definition is CLOB. No length is given, because in an Oracle database, a CLOB has no length.
`jdbc-precision`	Specifies the maximum number of digits stored in a column which represents a numeric type.
`jdbc-scale`	Specifies the number of digits stored to the right of the decimal point in a column that represents a floating point number.
`jdbc-nullable`	Specifies whether the column generated for the CMP field allows null values.

Example

```
<schema-generator-properties>
   <property>
         <name>Employee.firstName.jdbc-type</name>
         <value>char</value>
   </property>
   <property>
         <name>Employee.firstName.jdbc-maximum-length</name>
         <value>25</value>
   </property>
   <property>
         <name>use-unique-table-names</name>
         <value>true</value>
   </property>
</schema-generator-properties>
```

secondary-table

Specifies a bean's secondary table(s).

Superelements

"entity-mapping" on page 99 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–103 `secondary table` Subelements

Element	Required	Description
"table-name" on page 168	only one	Specifies the name of a database table.
"column-pair" on page 83	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 169 (`sun-acc.xml`)

Subelements

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

TABLE A–104 `security` Subelements

Element	Required	Description
"ssl" on page 157	only one	Specifies the SSL processing parameters.
"cert-db" on page 74	only one	Not implemented. Included for backward compatibility only.

security-role-mapping

Maps roles to users or groups in the currently active realm. See "Realm Configuration" in *Sun Java System Application Server 9.1 Developer's 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 161 (`sun-application.xml`), "sun-web-app" on page 164 (`sun-web.xml`), "sun-ejb-jar" on page 163 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–105 `security-role-mapping` Subelements

Element	Required	Description
"role-name" on page 148	only one	Contains the `role-name` in the `security-role` element of the corresponding Java EE deployment descriptor file.
"principal-name" on page 134	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 103	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 133 (`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 153 (`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 153 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`);
"webservice-endpoint" on page 176 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

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

TABLE A–106 `service-qname` subelements

Element	Required	Description
"namespaceURI" on page 128	only one	Specifies the namespace URI.
"localpart" on page 115	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 164 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`),
"sun-application-client" on page 161 (`sun-application-client.xml`)

Subelements

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

TABLE A–107 `service-ref` subelements

Element	Required	Description
"service-ref-name" on page 154	only one	Specifies the web service reference name relative to `java:comp/env`.
"port-info" on page 133	zero or more	Specifies information for a port within a web service reference.
"call-property" on page 73	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 177	zero or one	Specifies a valid URL pointing to a final WSDL document.
"service-impl-class" on page 152	zero or one	Specifies the name of the generated service implementation class.
"service-qname" on page 153	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 153 (`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 164 (`sun-web.xml`)

Subelements

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

TABLE A–108 `servlet` Subelements

Element	Required	Description
"servlet-name" on page 155	only one	Contains the name of a servlet, which is matched to a `servlet-name` in `web.xml`.
"principal-name" on page 134	zero or one	Contains a principal (user) name in the current realm.
"webservice-endpoint" on page 176	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 176 (`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 72, "servlet" on page 154 (`sun-web.xml`)

Subelements

none - contains data

session-config

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

Superelements

"sun-web-app" on page 164 (`sun-web.xml`)

Subelements

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

TABLE A–109 `session-config` Subelements

Element	Required	Description
"session-manager" on page 156	zero or one	Specifies session manager configuration information.
"session-properties" on page 156	zero or one	Specifies session properties.
"cookie-properties" on page 87	zero or one	Specifies session cookie properties.

session-manager

Specifies session manager information.

Superelements

"session-config" on page 155 (`sun-web.xml`)

Subelements

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

TABLE A–110 `session-manager` Subelements

Element	Required	Description
"manager-properties" on page 117	zero or one	Specifies session manager properties.
"store-properties" on page 158	zero or one	Specifies session persistence (storage) properties.

Attributes

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

TABLE A–111 `session-manager` Attributes

Attribute	Default	Description
`persistence-type`	`memory`	(optional) Specifies the session persistence mechanism. Allowed values are `memory`, `file`, and `replicated`. If HADB is installed and you have selected the enterprise profile, you can also specify `ha`. For production environments that require session persistence, use `ha`.

session-properties

Specifies session properties.

Superelements

"session-config" on page 155 (`sun-web.xml`)

Subelements

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

TABLE A–112 `session-properties` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–113 `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 151 (`sun-acc.xml`)

Subelements

none

Attributes

The following table describes attributes for the SSL element.

TABLE A–114 ssl attributes

Attribute	Default	Description
cert-nickname	s1as	(optional) The nickname of the server certificate in the certificate database or the PKCS#11 token. In the certificate, the name format is *tokenname*:*nickname*. Including the *tokenname*: part of the name in this attribute is optional.
ssl2-enabled	false	(optional) Determines whether SSL2 is enabled.
ssl2-ciphers	none	(optional) A 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 68 (sun-ejb-jar.xml)

Subelements

none - contains data

store-properties

Specifies session persistence (storage) properties.

Superelements

"session-manager" on page 156 (`sun-web.xml`)

Subelements

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

TABLE A–115 `store-properties` Subelements

Element	Required	Description
"property (with attributes)" on page 135	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 A–116 `store-properties` Properties

Property	Default	Description
`directory`	*domain-dir*`/generated/jsp/j2ee-apps/`*app-name*`/`*app-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 application. Applicable only if the `persistence-type` attribute of the parent "session-manager" on page 156 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 application. ■ `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 156 element is `ha` or `replicated`.

If the `persistenceScope` store property is set to `modified-attribute`, a web application 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 133 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–117 `stub-property` subelements

Element	Required	Description
"name" on page 128	only one	Specifies the name of the entity.
"value" on page 173	only one	Specifies the value of the entity.

Properties

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

TABLE A–118 `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 Application 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 49.

Superelements

none

Subelements

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

TABLE A–119 `sun-application` Subelements

Element	Required	Description
"web" on page 175	zero or more	Specifies the application's web tier configuration.
"pass-by-reference" on page 130	zero or one	Determines whether EJB modules use pass-by-value or pass-by-reference semantics.
"unique-id" on page 172	zero or one	Contains the unique ID for the application.
"security-role-mapping" on page 151	zero or more	Maps a role in the corresponding Java EE XML file to a user or group.
"realm" on page 139	zero or one	Specifies an authentication realm.

sun-application-client

Defines the Application 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 61.

Superelements

none

Subelements

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

TABLE A–120 `sun-application-client` subelements

Element	Required	Description
"ejb-ref" on page 95	zero or more	Maps the absolute JNDI name to the `ejb-ref` in the corresponding Java EE XML file.
"resource-ref" on page 146	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Java EE XML file.
"resource-env-ref" on page 145	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Java EE XML file.
"service-ref" on page 153	zero or more	Specifies runtime settings for a web service reference.
"message-destination" on page 122	zero or more	Specifies the name of a logical message destination.
"message-destination-ref" on page 123	zero or more	Specifies the name of a physical message destination.
"java-web-start-access" on page 106	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 163 (`sun-cmp-mappings.xml`)

Subelements

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

TABLE A–121 `sun-cmp-mapping` Subelements

Element	Required	Description
"schema" on page 149	only one	Specifies the file that contains a description of the database schema.
"entity-mapping" on page 99	one or more	Specifies the mapping of a bean to database columns.

sun-cmp-mappings

Defines the Application 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 57.

Superelements

none

Subelements

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

TABLE A–122 `sun-cmp-mappings` Subelements

Element	Required	Description
"sun-cmp-mapping" on page 162	one or more	Specifies beans mapped to a particular database schema.

sun-ejb-jar

Defines the Application 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 52.

Superelements

none

Subelements

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

TABLE A–123 `sun-ejb-jar` Subelements

Element	Required	Description
"security-role-mapping" on page 151	zero or more	Maps a role in the corresponding Java EE XML file to a user or group.
"enterprise-beans" on page 97	only one	Describes all the runtime properties for an EJB JAR file in the application.

sun-web-app

Defines Application 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 49.

Superelements

none

Subelements

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

TABLE A–124 `sun-web-app` Subelements

Element	Required	Description
"context-root" on page 87	zero or one	Contains the web context root for the web application.
"security-role-mapping" on page 151	zero or more	Maps roles to users or groups in the currently active realm.
"servlet" on page 154	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 104	zero or more	Specifies a URL pattern for idempotent requests.
"session-config" on page 155	zero or one	Specifies session manager, session cookie, and other session-related information.
"ejb-ref" on page 95	zero or more	Maps the absolute JNDI name to the `ejb-ref` in the corresponding Java EE XML file.
"resource-ref" on page 146	zero or more	Maps the absolute JNDI name to the `resource-ref` in the corresponding Java EE XML file.
"resource-env-ref" on page 145	zero or more	Maps the absolute JNDI name to the `resource-env-ref` in the corresponding Java EE XML file.

TABLE A–124 `sun-web-app` Subelements *(Continued)*

Element	Required	Description
"service-ref" on page 153	zero or more	Specifies runtime settings for a web service reference.
"message-destination-ref" on page 123	zero or more	Specifies the name of a physical message destination.
"cache" on page 69	zero or one	Configures caching for web application components.
"class-loader" on page 76	zero or one	Specifies class loader configuration information.
"jsp-config" on page 108	zero or one	Specifies JSP configuration information.
"locale-charset-info" on page 113	zero or one	Deprecated. Use the `parameter-encoding` subelement of `sun-web-app` instead.
"parameter-encoding" on page 130	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 135	zero or more	Specifies a property, which has a name and a value.
"message-destination" on page 122	zero or more	Specifies the name of a logical message destination.
"webservice-description" on page 175	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 A–125 `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 A–126 `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 "virtual-server" in *Sun Java System Application Server 9.1 Administration Reference*. **Caution –** Setting this property to `true` on Windows systems exposes JSP source code.

TABLE A–126 `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 "virtual-server" in *Sun Java System Application Server 9.1 Administration Reference*.

`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.

TABLE A–126 `sun-web-app` Properties *(Continued)*

Property	Default	Description
`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 "virtual-server" in *Sun Java System Application Server 9.1 Administration Reference*.
`singleThreaded ServletPoolSize`	`5`	Specifies the maximum number of servlet instances allocated for each `SingleThreadModel` servlet in the web application.
`tempdir`	*domain-dir*`/generated/j2ee-apps/`*app-name* or *domain-dir*`/generated/j2ee-modules/`*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).

T

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 *Sun Java System Application Server 9.1 Developer's Guide*.

Superelements

"entity-mapping" on page 99, "secondary-table" on page 151 (`sun-cmp-mappings.xml`)

Subelements

none - contains data

target-server

Specifies the IIOP listener for the target server. Also specifies IIOP endpoints used for load balancing. If the Application Server instance on which the application client is deployed participates in a cluster, the Application Server finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

A listener or endpoint is in the form *host*:*port*, where the *host* is an IP address or host name, and the *port* specifies the port number.

Not used if the deprecated `endpoints` property is defined for load balancing. For more information, see "client-container" on page 77.

Note – Some topics in the documentation pertain to features that are available only in domains that are configured to support clusters. Examples of domains that support clusters are domains that are created with the cluster profile or the enterprise profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*.

Superelements

"client-container" on page 77 (`sun-acc.xml`)

Subelements

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

TABLE A–127 `target-server` subelements

Element	Required	Description
"description" on page 91	zero or one	Specifies the description of the target server.
"security" on page 151	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 A–128 `target-server` attributes

Attribute	Default	Description
`name`	none	Specifies the name of the application server instance accessed by the client container.

TABLE A–128 `target-server` attributes *(Continued)*

Attribute	Default	Description
`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 Admin Console. Click the Help button in the Admin 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 176 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

timeout

Specifies the "cache-mapping" on page 72 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 69 element.

Superelements

"cache-mapping" on page 72 (`sun-web.xml`)

Subelements

none - contains data

Attributes

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

TABLE A–129 `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 105 (`sun-ejb-jar.xml`)

Subelements

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

TABLE A–130 `transport-config` Subelements

Element	Required	Description
"integrity" on page 104	only one	Specifies if the target supports integrity-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
"confidentiality" on page 84	only one	Specifies if the target supports privacy-protected messages. The values are NONE, SUPPORTED, or REQUIRED.
"establish-trust-in-target" on page 100	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 99	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 176 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

U

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 161 (`sun-application.xml`), "enterprise-beans" on page 97 (`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 72 (`sun-web.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 92 (`sun-ejb-jar.xml`)

Subelements

none - contains data

V

value

Specifies the value of the entity.

Superelements

"call-property" on page 73, "stub-property" on page 160 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`); "property (with subelements)" on page 136 (`sun-ejb-jar.xml`)

Subelements

none - contains data

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 106 (`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 67 (`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.

W

web

Specifies the application's web tier configuration.

Superelements

"sun-application" on page 161 (`sun-application.xml`)

Subelements

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

TABLE A–131 `web` Subelements

Element	Required	Description
"web-uri" on page 175	only one	Contains the web URI for the application.
"context-root" on page 87	only one	Contains the web context root for the application.

web-uri

Contains the web URI for the application. Must match the corresponding element in the `application.xml` file.

Superelements

"web" on page 175 (`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 164 (`sun-web.xml`), "enterprise-beans" on page 97 (`sun-ejb-jar.xml`)

Subelements

The following table describes subelements for the `webservice-description` element.

TABLE A–132 `webservice-description` subelements

Element	Required	Description
"webservice-description-name" on page 176	only one	Specifies a unique name for the web service within a web or EJB module.
"wsdl-publish-location" on page 178	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 175 (`sun-web.xml`, `sun-ejb-jar.xml`)

Subelements

none - contains data

webservice-endpoint

Specifies information about a web service endpoint.

Superelements

"servlet" on page 154 (`sun-web.xml`), "ejb" on page 92 (`sun-ejb-jar.xml`)

Subelements

The following table describes subelements for the `webservice-endpoint` element.

TABLE A–133 `webservice-endpoint` subelements

Element	Required	Description
"port-component-name" on page 132	only one	Specifies a unique name for a port component within a web or EJB module.
"endpoint-address-uri" on page 96	zero or one	Specifies the automatically generated endpoint address.
"login-config" on page 117	zero or one	Specifies the authentication configuration for an EJB web service endpoint.
"message-security-binding" on page 124	zero or one	Specifies a custom authentication provider binding.
"transport-guarantee" on page 171	zero or one	Specifies that the communication between client and server is `NONE`, `INTEGRAL`, or `CONFIDENTIAL`.
"service-qname" on page 153	zero or one	Specifies the WSDL service element that is being referenced.
"tie-class" on page 170	zero or one	Specifies the automatically generated name of a tie implementation class for a port component.
"servlet-impl-class" on page 155	zero or one	Specifies the automatically generated name of the generated servlet implementation class.
"debugging-enabled" on page 89	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 135 (`sun-web.xml`) "property (with subelements)" on page 136 (`sun-ejb-jar.xml`)	zero or more	Specifies a property, which has a name and a value.

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 153 (`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 133 (`sun-web.xml`, `sun-ejb-jar.xml`, `sun-application-client.xml`)

Subelements

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

TABLE A–134 `wsdl-port` subelements

Element	Required	Description
"namespaceURI" on page 128	only one	Specifies the namespace URI.
"localpart" on page 115	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 175 (`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

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S