J2EE CA SPI Administrator's Guide

Sun™ ONE Application Server

Version 7

817-2254-10
March 2003

Contents

About This Guide

The Sun™ ONE Application Server 7 Java 2 Enterprise Edition Connector Architecture (J2EE CA) Service Provider Implementation (SPI) is what allows a J2EE connector to plug in to the application server. The documentation explains how to build, install, configure, and manage the SPI.

This preface contains information about the following topics:

Product Line Overview

Sun ONE Application Server 7 is a J2EE 1.3 specification-compatible application server which also supports emerging Java Web Services standards as well as standard HTTP server programming facilities. Three editions of the application server are offered to suit a variety of needs for both production and development environments:

Platform Edition

Platform Edition forms the core of the Sun ONE Application Server 7 product line. This free-for-production-use product offers a high-performance, small-footprint J2EE 1.3 specification-compatible runtime environment that is ideally suited for basic operational deployments, as well as for embedding in third-party applications. Web-services ready, the Platform Edition includes built-in technologies proven by the Sun ONE Web Server and Sun ONE Messge Queue products.

Platform Edition deployments are limited to single application server instances (i.e. single virtual machines for the Java platform ("Java virtual machine" or "JVM™")). Multi-tier deployment topologies are supported by the Platform edition, but the web server tier proxy does not perform load balancing. In Platform Edition, administrative utilities are limited to local clients only.

Platform Edition is integrated into Solaris 9.

Standard Edition

This is the edition that is the focus of this *Getting Started Guide*. The Standard Edition layers enhanced, remote-management capabilities on top of the Platform Edition. Enhanced management capabilities, remote command-line, and web-based administration are all included as part of the Standard Edition. This edition also includes the ability to partition web application traffic through a web server tier proxy. Standard Edition supports configuration of multiple application server instances (JVMs) per machine.

Enterprise Edition

Enterprise Edition enhances the core application server platform with greater high availability, load balancing and clustering capabilities suited for the most demanding J2EE-based application deployments. The management capabilities of the Standard Edition are extended in Enterprise Edition to account for multi-instance and multi-machine deployments.

Clustering support includes easy-to-configure groups of cloned application server instances to which client requests can be load balanced. Both external load balancers and load balancing web tier-based proxies are supported by this edition. HTTP session, stateful session bean instance and Java Message Service (JMS) resource failover are included in the Enterprise Edition. The patented “Always On” highly available database technology forms the basis for the HA persistence store in the Enterprise Edition.

For more product information, see the Sun ONE Application Server page on the Sun Microsystems web site.

Using the Documentation

The Sun ONE Application Server manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML) formats, at:

`http://docs.sun.com`

The following table lists tasks and concepts described in the Sun ONE Application Server manuals. The left column lists the tasks and concepts, and the right column lists the corresponding manuals.

Table 1 Sun ONE Application Server Documentation Roadmap

For information about	See the following
Late-breaking information about the software and the documentation	*Release Notes*
Supported platforms and environments	*Platform Summary*
Introduction to the application server, including new features, evaluation installation information, and architectural overview.	*Getting Started Guide*
Installing Sun ONE Application Server and its various components (sample applications, Administration interface, Sun ONE Message Queue).	*Installation Guide*
Creating and implementing J2EE applications that follow the open Java standards model on the Sun ONE Application Server 7. Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules.	*Developer’s Guide*

Table 1 Sun ONE Application Server Documentation Roadmap *(Continued)*

For information about	See the following
Creating and implementing J2EE applications that follow the open Java standards model for web applications on the Sun ONE Application Server 7. Discusses web application programming concepts and tasks, and provides sample code, implementation tips, and reference material.	*Developer's Guide to Web Applications*
Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun ONE Application Server 7. Discusses EJB programming concepts and tasks, and provides sample code, implementation tips, and reference material.	*Developer's Guide to Enterprise JavaBeans Technology*
Creating Web Services, RMI-IIOP, or other clients that access J2EE applications on the Sun ONE Application Server 7	*Developer's Guide to Clients*
J2EE features such as JDBC, JNDI, JTS, JMS, JavaMail, resources, and connectors	*Developer's Guide to J2EE Features and Services*
Creating custom NSAPI plugins	*Developer's Guide to NSAPI*
Performing the following administration tasks:	*Administrator's Guide*
• Using the Administration interface and the command line interface	
• Configuring server preferences	
• Using administrative domains	
• Using server instances	
• Monitoring and logging server activity	
• Configuring the web server plugin	
• Configuring the Java Messaging Service	
• Using J2EE features	
• Configuring support for CORBA-based clients	
• Configuring database connectivity	
• Configuring transaction management	
• Configuring the web container	
• Deploying applications	
• Managing virtual servers	

Table 1 Sun ONE Application Server Documentation Roadmap *(Continued)*

For information about	See the following
Editing server configuration files	*Administrator's Configuration File Reference*
Configuring and administering security for the Sun ONE Application Server 7 operational environment. Includes information on general security, certificates, and SSL/TLS encryption. HTTP server-based security is also addressed.	*Administrator's Guide to Security*
Configuring and administering service provider implementation for J2EE CA connectors for the Sun ONE Application Server 7. Includes information about the Administration Tool, DTDs and provides sample XML files.	*J2EE CA Service Provider Implementation Administrator's Guide*
Migrating your applications to the new Sun ONE Application Server 7 programming model from the Netscape Application Server version 2.1, including a sample migration of an Online Bank application provided with Sun ONE Application Server	*Migration Guide*
Using Sun ONE Message Queue.	The Sun ONE Message Queue documentation at: http://docs.sun.com/?p=/coll/S1_MessageQueue_30

Documentation Conventions

This section describes the types of conventions used throughout this guide:

- General Conventions
- Conventions Referring to Directories

General Conventions

The following general conventions are used in this guide:

- **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.
- **URLs** are given in the format:

http://*server.domain*/*path*/*file*.html

In these URLs, *server* is the server name where applications are run; *domain* is your Internet domain name; *path* is the server's directory structure; and *file* is an individual filename. Italic items in URLs are placeholders.

- **Font conventions** include:
  - The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
  - *Italic* type is used for code variables.
  - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
  - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.
- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in "Conventions Referring to Directories" on page 11.

  By default, the location of *install_dir* on **most** platforms is:
  - Solaris 8 non-package-based Evaluation installations:

    `user's home directory/sun/appserver7`
  - Solaris unbundled, non-evaluation installations:

    `/opt/SUNWappserver7`
  - Windows, all installations:

    `C:\Sun\AppServer7`

  For the platforms listed above, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See "Conventions Referring to Directories" on page 11 for exceptions and additional information.
- **Instance root directories** are indicated by *instance_dir* in this document, which is an abbreviation for the following:

  `default_config_dir``/domains/`*`domain/instance`*
- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris 8 and 9 package-based installation and the Solaris 9 bundled installation, the application server files are spread across several root directories. These directories are described in this section.

- **For Solaris 9 bundled installations**, this guide uses the following document conventions to correspond to the various default installation directories provided:
  - *install_dir* refers to `/usr/appserver/`, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
  - *default_config_dir* refers to `/var/appserver/domains`, which is the default location for any domains that are created.
  - *install_config_dir* refers to `/etc/appserver/config`, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.
- **For Solaris 8 and 9 package-based, non-evaluation, unbundled installations**, this guide uses the following document conventions to correspond to the various default installation directories provided:
  - *install_dir* refers to `/opt/SUNWappserver7`, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
  - *default_config_dir* refers to `/var/opt/SUNWappserver7/domains`which is the default location for any domains that are created.
  - *install_config_dir* refers to `/etc/opt/SUNWappserver7/config`, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

Product Support

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support web site at:

  `http://www.sun.com/supportraining/`
- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

How This Guide Is Organized

The *Sun ONE Application Server J2EE CA SPI Administrator's Guide* provides the information you need to understand, set up, and administer all aspects of the Sun ONE Application Server J2EE CA SPI software.

This guide contains the following documentation components:

- Chapter 1, "Introduction to J2EE Connector Architecture"
- Chapter 2, "Assembling and Deploying the Connector"
- Chapter 3, "Administering the Resource Adapter"
- Chapter 4, "The Sample Connector and its Use in the Sample Application"
- Appendix A, "Troubleshooting"

Finally a Glossary and Index are provided.

NOTE	Throughout this manual, all Unix-specific descriptions apply to the Linux operating system as well, except where Linux is specifically mentioned.

Chapter 1

Introduction to J2EE Connector Architecture

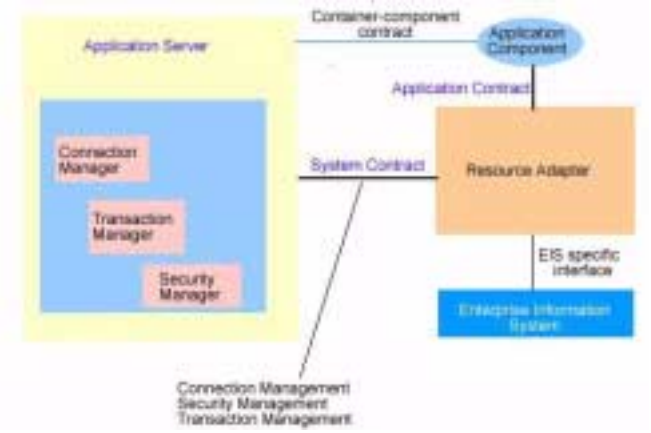
This guide is written for administrators that need to deploy and use J2EE CA resource adapters with the Sun ONE Application Server. This guide provides instructions for assembling and deploying J2EE CA resource adapters.

This chapter describes the following topics:

J2EE CA Overview

The J2EE Connector Architecture defines a standard architecture for connecting the J2EE platform to heterogeneous EISs. Examples of EISs include ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language. By defining a a set of scalable, secure, and transactional mechanisms, the J2EE Connector Architecture enables the integration of EISs with application servers and enterprise applications.

The J2EE Connector architecture enables an EIS vendor to provide a standard resource adapter for its EIS. The resource adapter plugs into an application server, providing connectivity between the EIS, the application server, and the enterprise application. If an application server vendor has extended its system to support the J2EE Connector architecture, it is assured of seamless connectivity to multiple EISs. An EIS vendor needs to provide just one standard resource adapter that has the capability to plug in to any application server that supports the J2EE Connector architecture. Multiple resource adapters are pluggable into an application server. This capability enables application components deployed on the application server to access the underlying EISs. The following diagram illustrates the J2EE Connector architecture.



Application Server - implements the J2EE CA server side contracts, such as transaction, pooling (connection) and security.

Resource Adapter- hooks into the application server using the server side contracts. You can use the Sun One Connector Builder to generate a resource adapter.

Enterprise Information System - is the backend system that connects to the Application Server via the system contracts.

Application Component - can be a server-side component, such as an EJB, MDB, JSP, or Servlet, that is deployed, managed, and executed on an application server. It can also be a component executed on the web-client tier but made available to the web-client by an application server. Examples of the latter type of application component include a Java applet and DHTML page.

Chapter 2

Assembling and Deploying the Connector

This chapter describes the contents of the Sun ONE Application Server connector module and how this module is assembled separately or together in an application.

The Sun ONE Application Server connector module includes J2EE standard elements and Sun ONE Application Server specific elements. Only Sun ONE Application Server specific elements are described in detail in this chapter.

This chapter describes the following topics:

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 J2EE-compliant application server. A package can be classified either as a module or as a full-fledged application. This section covers the following topics:

- J2EE Standard Descriptors
- Sun ONE Application Server Descriptors
- Naming Standards
- Runtime Environments
- Classloaders
- Sample Applications

Modules

A J2EE module is a collection of one or more J2EE components of the same container type with two deployment descriptors of that type. One descriptor is J2EE standard, the other is Sun ONE Application Server specific. The Connector Module can be described as follows:

- **Resource RAR File:** The RAR files apply only to J2EE CA connectors. Each Sun ONE Application Server connector has a `sun-ra.xml` file and a J2EE application deployment descriptor `ra.xml` file.

Package definitions must be used in the source code of all modules so the classloader 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 J2EE server reads this information and acts accordingly.

The connector module has a Sun ONE Application Server deployment descriptor and a J2EE deployment descriptor. The Sun ONE Application Server Administration interface uses the deployment descriptors to deploy the application components and to register the resources with the Sun ONE Application Server.

An application consists of one or more modules, a Sun ONE Application Server deployment descriptor, and a J2EE application deployment descriptor. All modules are assembled, using the Java ARchive (`.jar`) file format, into one file with an extension of `.ear`.

J2EE Standard Descriptors

The J2EE platform provides assembly and deployment facilities. These facilities use JAR files as the standard package for components and applications, and XML-based deployment descriptors for customizing parameters.

The J2EE standard deployment descriptors are described in the J2EE specification, v1.3. For more information on J2EE standard deployment descriptors, see the following specifications:

- *Java 2 Enterprise Edition, J2EE Connector Architecture Specification v1.0, Chapter 10: Packaging and Deployment.*

You can find specifications at the following URL:

`http://java.sun.com/products/`

Sun ONE Application Server Descriptors

Sun ONE Application Server uses additional deployment descriptors for configuring features specific to the Sun ONE Application Server.

NOTE The Sun ONE Application Server deployment descriptors must have 600 level access privileges on Unix systems.

The DTD schema files for all the Sun ONE Application Server deployment descriptors are located in the `install_dir/lib/dtds` directory.

Naming Standards

Names of deployed connector RAR modules must be unique in the Sun ONE Application Server. Using a Java package-like naming scheme is recommended for module filenames. 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 Sun ONE Application Server, but to other J2EE application servers as well.

Runtime Environments

Whether you deploy a component as a stand-alone module or as an application, deployment affects both the file system and the server configuration. See the "*Sun ONE Application Server J2EE CA SPI Developer's Guide*" for more information.

Classloaders

In a Java Virtual Machine (JVM), the classloaders dynamically load a specific java class file needed for resolving a dependency. For example, when an instance of `java.util.Enumeration` needs to be created, one of the classloaders loads the relevant class into the environment. See the "*Sun ONE Application Server J2EE CA SPI Developer's Guide*" for more information.

Sample Applications

Sample applications that you can examine and deploy are included in Sun ONE Application Server, in the `install_dir/samples/j2ee` directory. Each sample has its own documentation.

Selects component files that match specified parameters. When fileset is included as a subelement, the name and contextroot attributes of the containing element must use their default values for each file in the fileset. For more information, see:

`http://jakarta.apache.org/ant/manual/CoreTypes/fileset.html`

Assembling a J2EE CA Resource Adapter

This section provides some brief pointers for assembling J2EE CA resource adapters.

The following two XML connector files are required for deploying a connector to the application server.

- `sun-ra.xml`
- `ra.xml`

The `ra.xml` file is based on the J2EE CA specification and is packaged with the connector. The `sun-ra.xml` file contains Sun ONE Application Server specific information.

To assemble a connector RAR module

1. Create a working directory, and copy the contents of your module into it.
2. Create two deployment descriptor files with these names: `sun-ra.xml` and `ra.xml` in the META_INF directory.
3. Execute this command to create the RAR file:

   jar -cvf module_name.rar *

Deploying Modules and Applications

This section describes the way to deploy J2EE connector module to the Sun ONE Application Server.

Deployment IDs and Errors

A unique ID is generated in the `server.xml` file when you deploy an application or module. Do not change this ID. During deployment, the server detects any ID collisions and does not load an application or module having a non-unique ID. Messages are logged in the server log when this happens.

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.

For details about `server.xml`, see the *Sun ONE Application Server Administrator's Configuration File Reference*.

The Deployment Life Cycle

After an application is initially deployed, it may be modified and reloaded, redeployed, disabled, re-enabled, and finally undeployed (removed from the server). This section covers the following topics related to the deployment life cycle:

- Dynamic Deployment
- Disabling a Resource Adapter
- Dynamic Reloading

Dynamic Deployment

You can deploy, redeploy, and undeploy an application or module without restarting the server. This is called dynamic deployment.

When an application or module is redeployed, some file system content and some Application Server settings are not overwritten or removed. This can lead to older settings remaining in effect after a redeployment. To perform a clean redeployment, undeploy the application or module before redeploying it. In addition, whenever a redeployment is done, the sessions at that transit time become invalid. The client must restart the session.

Disabling a Resource Adapter

You can disable a deployed resource adapter without removing it from the server. Each application and module has an `enabled` attribute in the `server.xml` file and a corresponding option in the Administration interface, which you can change. For details about `server.xml`, see the *Sun ONE Application Server Administrator's Configuration File Reference*.

Dynamic Reloading

If dynamic reloading is enabled, you do not have to redeploy an application or module when you change its code. All you have to do is copy the changed 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 is useful in a development environment, because it allows code changes to be tested quickly. Dynamic reloading is not recommended for a production environment, however, because it may degrade performance. In addition, whenever a reload is done, the sessions at that transit time become invalid. The client must restart the session.

The deployment descriptor files in the application are not reloaded automatically. If you change them, you must redeploy the application.

To enable dynamic reloading, you can do one of the following:

- Use the Administration Tool interface:
  - **a.** Open the Applications component under your server instance
  - **b.** Go to the Applications page
  - **c.** Check the Reload Enabled box to enable dynamic reloading.

d. Enter a number of seconds in the Reload Poll Interval field to set the interval at which applications and modules are checked for code changes and dynamically reloaded.

e. Click on the Save button.

- Edit the following attributes of the `server.xml` file's `applications` element:
  - `dynamic-reload-enabled="true"` enables dynamic reloading.
  - `dynamic-reload-poll-interval-in-seconds` sets the interval at which applications and modules are checked for code changes and dynamically reloaded.

  For details about `server.xml`, see the *Sun ONE Application Server Administrator's Configuration File Reference*.

In addition, to load new servlet files, reload EJB related changes, or reload deployment descriptor changes, you must do the following:

- Create an empty file named `.reload` at the root of the deployed application:

  instance_dir`/applications/j2ee-apps/`*app_name*`/.reload`

  or individually deployed module:

  instance_dir`/applications/j2ee-modules/`*module_name*`/.reload`

- Explicitly update the `.reload` file's timestamp (`touch .reload` in Unix) each time you make the above changes.

For JSPs, changes are reloaded automatically at a frequency set in the `reload-interval` property of the `jsp-config` element in the `sun-web.xml` file. To disable dynamic reloading of JSPs, set `reload-interval="-1"`.

Deploying a J2EE CA Resource Adapter

You can deploy a connector module using the following tools:

- asadmin deploy
- The Administration Interface

asadmin deploy

The `asadmin deploy` command deploys a RAR file. To deploy a stand-alone connector module, specify `--type connector`.The following command deploys a RAR stand-alone module:

```
asadmin deploy -- connector--instance inst1 -name connector_name rar
filename
```

To undeploy a connector module, use this command:

```
asadmin undeploy -- connector--instance inst1 -name connector_name
```

The Administration Interface

You can also use the Administration interface to deploy a connector module.

To Deploy a Connector Module

1. Open the Applications component under your server domain.
2. Go to the Connector Modules page.
3. Click on the Deploy button.
4. Enter the following information

   File path - The file path for the connector module .rar file.
5. Click OK.
6. Enter the connector name. Click OK

Access to Shared Frameworks

When J2EE applications and modules use shared framework classes (such as components and libraries) the classes can be put in the path for the System Classloader or the Common Classloader 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 about the system classloader, see “Classloaders” on page 18.

Security Management

Connectors are integrated in the overall J2EE security scheme. However, because the EIS usually has its own security system, a mechanism is required to map J2EE security principal to EIS principals. The J2EE SPI supports the following two mechanisms:

- principal mapping
- configured identity

Principal Mapping

Principal mapping sets up the mapping between J2EE principals EIS principals. The sun-ra.xml contains this mapping. For a description and sample see "Elements in the sun-ra.xml File" on page 26

Configured Identity

The configured identity mechanism allows you to set up and use a single user id to login to the backend for all J2EE principals.

J2EE CA currently supports password authentication.

The Connector Deployment Descriptor Files

Sun ONE Application Server connectors include two deployment descriptor files:

- A J2EE standard file (`ra.xml`), described in the Java 2 Enterprise Edition, J2EE Connector Architecture Specification v1.0, Chapter 10 "Packaging and Deployment".
- A Sun ONE Application Server specific file (`sun-ra.xml`), described in this section.

This section covers the following topics:

The sun-connector_1_0-0.dtd

The `sun-connector_1_0-0.dtd` file defines the various elements that the `sun-ra.xml` file can contain and the subelements and attributes these elements can have. The `sun-connector_1_0-0.dtd` file is located in the `install_dir/lib/dtds` directory.

NOTE Do not edit the `sun-connector_1_0-0.dtd` file. Its contents change only with new versions of Sun ONE Application Server.

NOTE The `sun-connector_1_0-0.dtd` interface is unstable. An unstable interface may be experimental or transitional, and hence may change incompatibly, beremoved, or be replaced by a more stable interface in the next release.

For general information about DTD files and XML, see the XML specification at:

`http://www.w3.org/TR/REC-xml`

Each element in a DTD file (and the XML file for which it specifies the schema) can contain the following:

- Subelements
- Data
- Attributes

Subelements

Elements can contain subelements. For example, the following code defines the `sun-connector` element.

```
<!ELEMENT sun-connector (resource-adapter, role-map?)>
```

The `ELEMENT` line specifies that a `sun-connector` element can contain `(resource-adapter, role-map?)` subelements.

The following table shows how the ending characters of the subelements determine the requirement rules for the subelements. The left column lists the subelement ending character, and the right column lists the corresponding requirement rule.

Table 2-1 Requirement Rules for Subelements

Subelement Ending Character	Requirement Rule
*	Can contain *zero or more* of this subelement.
?	Can contain *zero or one* of this subelement.
+	Must contain *one or more* of this subelement.
(none)	Must contain *only one* of this subelement.

If an element cannot contain other elements, you see `EMPTY` or `(#PCDATA)` instead of a list of element names in parentheses.

Data

Some elements contain data instead of subelements. These elements have definitions of the following format:

`<!ELEMENT` *element-name* `(#PCDATA)>`

For example:

```
<!ELEMENT description (#PCDATA)>
```

In the `sun-ra.xml` file, white space is treated as part of the data in a data element. Therefore, there should be no extra white space before or after the data delimited by a data element. For example:

```
<principal user-name="keren"></principal>
```

The following conventions apply to all J2EE deployment descriptor elements unless indicated otherwise.

In elements that contain PCDATA, leading and trailing white space in the data may be ignored.

In elements whose value is an "enumerated type", the value is case sensitive.

In elements that specify a pathname to a file within the same JAR file, relative filenames (i.e., those not starting with "/") are considered relative to the root of the JAR file's namespace. Absolute filenames (i.e., those starting with "/") also specify names in the root of the JAR file's namespace. In general, relative names are preferred. The exception is .war files where absolute names are preferred for consistency with the servlet API.

Attributes

Elements that have ATTLIST lines contain attributes.

Elements in the sun-ra.xml File

This section describes the following XML elements in the sun-connector_1_0-0.dtd sun-ra.xml files:

- sun-connector
- resource-adapter
- role-map
- map-element
- principal
- backend-principal
- description
- property

sun-connector

Defines Sun ONE Application Server specific sun-connector element and is the root element of the deployment descriptor for the resource adapter; there can only be one sun-connector element in a sun-ra.xml file.

Subelements

The following table describes subelements for the sun-connector element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

Table 2-2 sun-connector subelements

Element	Required	Description
resource-adapter		consists of the jndi-name and pooling configuration attributes
role-map		contains security mapping information

NOTE To avoid collisions with names of other enterprise resources in JNDI, and to avoid portability problems, all names in a Sun ONE Application Server application should begin with the string java:comp/env.

resource-adapter

Defines the pool sizing and JNDI - lookup name.

Attributes

The following table describes attributes for the `resource-adapter` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes the attribute.

Table 2-3 `resource adapter` attributes

Attribute	Default	Description
`jndi-name`		The jndi-name is the name by which, this resource adapter will appear in JNDI namespace. See following note.
`max-pool-size`	32	maximum number of connections to the EIS
`steady-pool-size`	4	minimum number of connections to be maintained
`max-wait-in-millis`	10000	If a connection is not available, the caller will have to wait this long, before a connection is created. A value of 0 implies, if there is no connection available an exception will be thrown. If the pool is completely utilized and the timer expires, an exception will be delivered to the application
`idle-timeout-in-seconds`	`1000`	A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connection with expired timeout.

NOTE The jndi-name written here is used as the jndi lookup name for all applications that use this connector. The jndi name should be unique. For example the jndi-name *comet* should be referred to as java:comp / env / eis / comet in the jndi namespace lookup.

Subelements

The following table describes subelements for the `resource-adapter` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

Table 2-4 `resource-adpater` subelements

Element	Required	Description
`description`		Describes the resource adapter
`property`		Defines the syntax for supplying properties as name value pairs.

description

Describes the resource adapter.

Subelements

None

property

Defines the syntax for supplying properties as name value pairs.

Subelements

None

Attributes

The following table describes attributes for the `property` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes the attribute.

Table 2-5 `property` attributes

Attribute	Default	Description
`name`		Name of the property
`value`		Contains the value of the property.

role-map

Defines the mapping from the principal received during Servlet / EJB authentication, to credentials accepted by the EIS. This mapping is optional. The map consists of several 2-tuples.

Subelements

The following table describes subelements for the `role-map` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

`role map` subelements

Element	Default	Description
`description`		Describes the backend that you are mapping to.
`map-element`		Describes the mapping from the principal to the backend principal.

Attributes

The following table describes attributes for the `role-map` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes the attribute.

Table 2-6 `role-map` attributes

Attribute	Default	Description
`map-id`		Describes the ID for the mapping.

map-element

Defines the map-element. It is possible to map multiple (server) principal to the same backend principal.

Subelements

The following table describes subelements for the `map-element`element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

Table 2-7 `map-element` subelements

Element	Default	Description
`principal`		Principal of the Servlet and EJB client
`backend-principal`		Backend principal of the EIS.

principal

Defines the principal of the Servlet and EJB client.

Subelements

The following table describes subelements for the `prinicpal` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

Table 2-8 `principal` subelements

Element	Default	Description
`description`		Describes the principal

Attribute

The following table describes attributes for the `principal` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes the attribute.

Table 2-9 `prinicpal` attributes

Attribute	Default	Description
`user-name`		Contains the user name.

backend-principal

Defines the backend EIS principal.

Subelements

None

Attributes

The following table describes attributes for the `backend-principal` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes the attribute.

`backend-prinicpal` attributes

Attribute	Default	Description
`user-name`		Contains the backend user name.
`password`		Contains the backend password.
`credential`		Contains the type of credential.

description

Defines the resource adapter.

Subelements

None

Sample Application XML FIles

The `sun-ra.xml` file needs to be customized to deploy it. Code Example provides an example of the code and a description of each parameter and item that needs to be customized.

This section includes the following:

- Sample sun-ra.xml File
- Sample ra.xml File

Sample sun-ra.xml File

The following is a code example of the `sun-ra.xml` file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-connector PUBLIC "-//Sun Microsystems, Inc.//DTD Sun
ONE Application Server 7.0 Connector 1.0//EN"
"http://www.sun.com/software/sunone/appserver/dtds/sun-connector_1_
0-0.dtd">

<sun-connector>

    <resource-adapter jndi-name="Comet" max-pool-size="20"
steady-pool-size="10" max-wait-time-in-millis="300000"
idle-timeout-in-seconds="5000">

    </resource-adapter>

    <role-map map-id="mainframe">

         <map-element>

           <principal user-name="keren"></principal>

              <backend-principal user-name="pazit" password="tulip"
credential="credential">

              </backend-principal>

          </map-element>

        </role-map>

    </sun-connector>
```

Sample ra.xml File

The following is an example of a sample `ra.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>


<!--

   Copyright 2002 Sun Microsystems, Inc. All rights reserved.

 -->


<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector
1.0//EN' '

http://java.sun.com/dtd/connector_1_0.dtd'>


<connector>
```

```
    <display-name>CometResourceAdapter</display-name>
    <vendor-name>sun</vendor-name>
    <spec-version>1.0</spec-version>
    <eis-type>Comet</eis-type>
    <version>1.0</version>
    <resourceadapter>

<managedconnectionfactory-class>samples.connectors.simple.CometMana
gedConnectionFactory</managedconnectionfactory-class>

<connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>

<connectionfactory-impl-class>samples.connectors.simple.CometConnec
tionFactory</connectionfactory-impl-class>

<connection-interface>javax.resource.cci.Connection</connection-int
erface>

<connection-impl-class>samples.connectors.simple.CometConnection</c
onnection-impl-class>
        <transaction-support>NoTransaction</transaction-support>
        <config-property>
            <config-property-name>Host</config-property-name>

<config-property-type>java.lang.String</config-property-type>

<config-property-value>localhost</config-property-value>
        </config-property>
        <config-property>
            <config-property-name>Port</config-property-name>

<config-property-type>java.lang.String</config-property-type>
            <config-property-value>8020</config-property-value>
```

```
        </config-property>

        <authentication-mechanism>

<authentication-mechanism-type>BasicPassword</authentication-mechan
ism-type>

<credential-interface>javax.resource.security.PasswordCredential</c
redential-interface>
        </authentication-mechanism>
        <reauthentication-support>false</reauthentication-support>
    </resourceadapter>
</connector>
```

Preparing the .rar File for Deployment

The .rar file must contain the following:

- A deployment descriptor file that contains standard deployment information that is defined in the connector_1-0.dtd file.

  The deployment descriptor must be stored with the name META-INF/ra.xml in the .rar file.

- The Sun ONE Application Server connector `sun-ra.xml` file that specifies additional Application Server specific deployment information.

  The `sun-ra.xml` must be stored with the name META-INF/sun-ra.xml in the .rar file.

- The Java interfaces, implementation and utility classes, required by the resource adapter should be packaged as one or more .jar files as part of the resource adapter module.

- The platform-dependent libraries required by the resource adapter should also be packaged with the resource adapter module.

Directory Structure for the .rar File

The following illustrates a listing of files in a sample resource adapter module:

- /META-INF/ra.xml
- /META-INF/sun-ra.xml
- /readme.html
- /ra.jar
- /client.jar
- /win.dll
- /solaris.so

In the above example, ra.xml is the deployment descriptor, sun-ra.xml is the Sun specific deployment descriptor. The ra.jar and client.jar contain Java interfaces and implementation classes for the resource adapter. The win.dll and solaris. so are examples of native libraries.

Administering the Resource Adapter

The J2EE CA SPI Implementation administrative tasks are explained inthis chapter.

This chapter describes the following topics:

Overview

The J2EE CA spec describes deployment of connectors but does not address the issue of administration of resource adapters.

After deploying a resource adapter, you may need to modify the parameters of an existing resource adapter.

Administrative Tasks

You can modify all the resource adapter's parameters, such as pooling, configuration, and security using either one of the following methods:

- Edit / modify the ra.xml and / or sun-ra.xml file in the .rar file and redeploy the resource adapter.
- Edit / modify the deployed ra.xml and / or sun-ra.xml file located in `<AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/applications/j2ee_modules/<connector_name>/META-INF` and restart the server.

  For more information see *The Connector Deployment Descriptor Files* in *Administering the Resource Adapter*.

Creating Multiple Instances of a Resource Adapter

If you have multiple backend systems of the same type, for example 5 different CICS systems, you need to deploy the resource adapter for each backend system.

Be certain that each resource adapter has a unique application server name, jndi name and connection parameters specific to the backend.

Security Configuration

According to the J2EE CA specification, the resource adapter needs to have the necessary permissions to read private credentials. AS7 ships with a default server (security) policy that allows the default user, ANONYMOUS, to read private credentials. If you are planning on using the resource adapter with container managed security and users other then ANONYMOUS, you need to modify the `server.policy` file so that the resource adapter will be able to read the private credentials for the connector users. See *Sun ONE Application Server J2EE CA SPI Developer's Guide* for more information concerning server policy.

There are two options that can be used depending on the number of users:

- Numerous users
- Limited Number of Users

Numerous users

Add the following permission to the server.policy

```
grant codeBase
"file:/AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/
applications/j2ee-modules/<Connector_directory>/-"{

 permission javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential

com.sun.enterprise.security.PrincipalImpl \"*\"", "read";

};
```

This will allow just the connector code to read the private credentials for any user.

Limited Number of Users

If there will be only a limited set of users of the connector, permission can be restricted to only those users as shown in the following:

Add the following permission to the server.policy for every user:

```
grant codeBase
"file:/AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/
applications/j2ee-modules/<Connector_directory>/-"{

 permission javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential

com.sun.enterprise.security.PrincipalImpl \"<user_name>\"", "read";

};
```

The Sample Connector and its Use in the Sample Application

This chapter describes the following topics:

Overview About Using the Comet Sample Application

The Comet Sample Connector can be used to illustrate how a J2EE CA compliant connector can be deployed and then operate with a J2EE CA compliant application server. The sample contains the following components:

- Sample J2EE CA Connector
- Sample Application
- Sample Backend

Compiling and Assembling the Sample Application

To easily recompile, assemble and deploy the application, see `J2EE Application Assembler` in the `Studio 4,Enterprise Edition for Java` JavaHelp for details on using a build facility to quickly perform these tasks.

For example, to rebuild the entire application from scratch, follow these steps:

1. Compile and Assemble Application

   Execute "build" under comet/src/

   The default target core will be executed to rebuild the WAR, JAR and EAR files.

2. Redeploy the Application.

   Execute "build deploy" under comet/src/

3. Restart the Application Server.

   An application server restart will be necessary if you've modified deployment descriptors. For servlet and/or JSP modification, no restart is necessary.

4. To clean the sample application project area, execute "build clean".

The Sample Connector

The `comet.rar` file contains the sample connector files that are archived into a .rar file according to the connector specification architecture. The file is located in `[sun7 installation directory]/sun70/samples/j2ee/connector/assemble/comet.rar`.

Description of Sample Connector (comet.rar)

The following describes the files in the Comet Connector Sample:

- META-INF/ra.xml — contains standard deployment information.
- comet.jar— contains JAVA classes and error messages file
- META-INF/sun-ra.xml — contains additional deployment information beyond the standard deployment descriptor.

META-INF/ra.xml

The Comet Sample ra.xml file contains the Host and Port configuration properties.

Their values are set to:

Host: localhost

Port: 8020

These values may be modified before deployment.

The following code displays the sample ra.xml file.

The following is an example of a sample `ra.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>


<!--
   Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 -->


<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector
1.0//EN' '
http://java.sun.com/dtd/connector_1_0.dtd'>


<connector>
    <display-name>CometResourceAdapter</display-name>
    <vendor-name>sun</vendor-name>
    <spec-version>1.0</spec-version>
    <eis-type>Comet</eis-type>
    <version>1.0</version>
    <resourceadapter>


<managedconnectionfactory-class>samples.connectors.simple.CometMana
gedConnectionFactory</managedconnectionfactory-class>


<connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>
```

```
<connectionfactory-impl-class>samples.connectors.simple.CometConnec
tionFactory</connectionfactory-impl-class>

<connection-interface>javax.resource.cci.Connection</connection-int
erface>


<connection-impl-class>samples.connectors.simple.CometConnection</c
onnection-impl-class>
        <transaction-support>NoTransaction</transaction-support>
        <config-property>
            <config-property-name>Host</config-property-name>

<config-property-type>java.lang.String</config-property-type>

<config-property-value>localhost</config-property-value>
        </config-property>
        <config-property>
            <config-property-name>Port</config-property-name>

<config-property-type>java.lang.String</config-property-type>
            <config-property-value>8020</config-property-value>
        </config-property>


        <authentication-mechanism>


<authentication-mechanism-type>BasicPassword</authentication-mechan
ism-type>


<credential-interface>javax.resource.security.PasswordCredential</c
redential-interface>
        </authentication-mechanism>
        <reauthentication-support>false</reauthentication-support>
    </resourceadapter>
```

```
</connector>
```

comet.jar

The `comet.jar` file contains the Java interfaces, implementation and utility classes, required by the resource adapter and the error messages file. The filename is samples.connectors.simple.Messages.properties. The messages file may be modified to comply with the internationalization standards (I18N).

META-INF/sun-ra.xml

The `sun-ra.xml` contains additional deployment information as follows:

- Pooling Configuration
- Security Role Map
- JNDI Name

Pooling Configuration

The table below lists the pooling configuration attributes. The left most column lists the attribute; the middle column lists its default value; the right most column provides a description of the attribute.

Table 4-1 Pooling attributes

Attribute	Default	Description
`max-pool-size`	32	maximum number of connections to the EIS
`steady-pool-size`	4	minimum number of connections to be maintained
`max-wait-in-millis`	10000	If a connection is not available, the caller will have to wait this long, before a connection is created. A value of 0 implies, if there is no connection available an exception will be thrown. If the pool is completely utilized and the timer expires, an exception will be delivered to the application

Table 4-1 Pooling attributes

Attribute	Default	Description
idle-timeout-in-seconds	1000	A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connection that their timeout has expired.

Security Role Map

The security role map attributes are used to perform mapping from the principal received during Servlet/EJB authentication, to credentials accepted by the EIS.

It is possible to map multiple principals to the same backend principal.

JNDI Name

The jndi-name is used as the jndi lookup name for all applications that use this connector. The jndi name should be unique. For example the jndi-name *comet* should be referred to as `java:comp/env/eis/comet` in the jndi namespace lookup.

The following code displays the sample `sun-ra.xml` file.

The following is a code example of the `sun-ra.xml` file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-connector PUBLIC "-//Sun Microsystems, Inc.//DTD Sun
ONE Application Server 7.0 Connector 1.0//EN"
"http://www.sun.com/software/sunone/appserver/dtds/sun-connector_1_
0-0.dtd">

<sun-connector>

    <resource-adapter jndi-name="Comet" max-pool-size="20"
steady-pool-size="10" max-wait-time-in-millis="300000"
idle-timeout-in-seconds="5000">

    </resource-adapter>

    <role-map map-id="mainframe">

         <map-element>

           <principal user-name="keren"></principal>

              <backend-principal user-name="pazit" password="tulip"
credential="credential">
```

```
            </backend-principal>
        </map-element>
    </role-map>
</sun-connector>
```

Deploying the Sample Connector

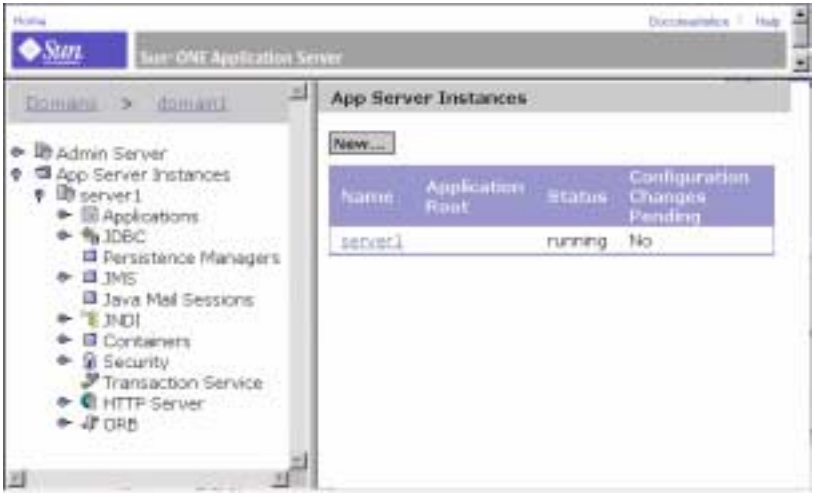
You need to deploy the sample connector to the application server so that the sample connector can be used.

To Deploy the Sample Connector

1. Start the Admin Server Tool.

NOTE The default instance name is server1

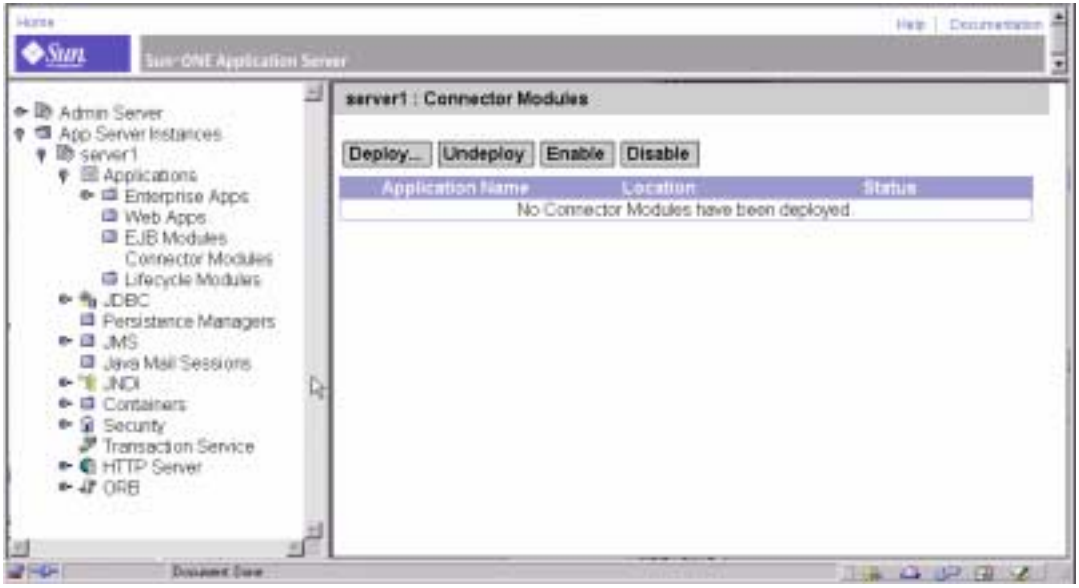
The following figure is displayed:



2. Select Application Server Instances-[instance name]-Applications > Connector Modules.

   The following figure is displayed:

.



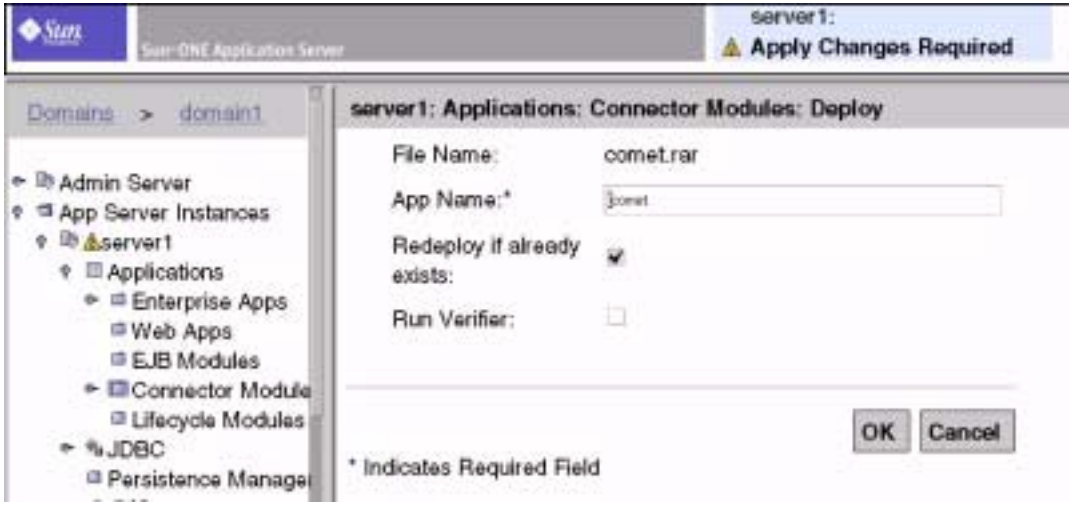
3. Click Deploy.
4. Browse to the location where the comet.rar file is located.

   The `comet.rar` file contains the sample connector files that are archived into a .rar file according to the connector specification architecture.

   The directory is:

   `[sun7 installation directory]/samples/connectors/simple`

5. Click OK. The following figure is displayed.

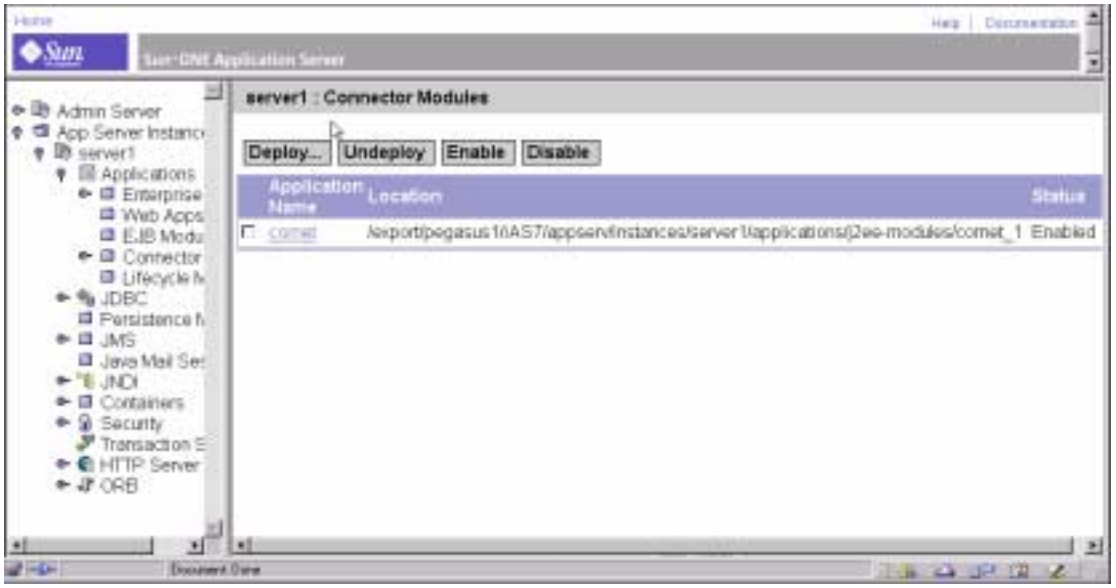


6. Insert the App Name: Comet

   Comet is the connector JNDI name that is used by the application to find the connector.

7. Click OK.

   The connector is added to the Connector Modules list. The following figure is displayed.



The Sample Application

The Comet Sample application operates the Comet connector that sends and receives messages from a backend system. You need to deploy the sample application and connector before you can use / operate it.

Directory Structure of the Sample Application

In general, the top-level directory of every sample application contains the following:

- src directory — contains source code for sample application
- docs — directory where all documentation is available
- assemble — contains the .EAR file that needs to be deployed to the application server

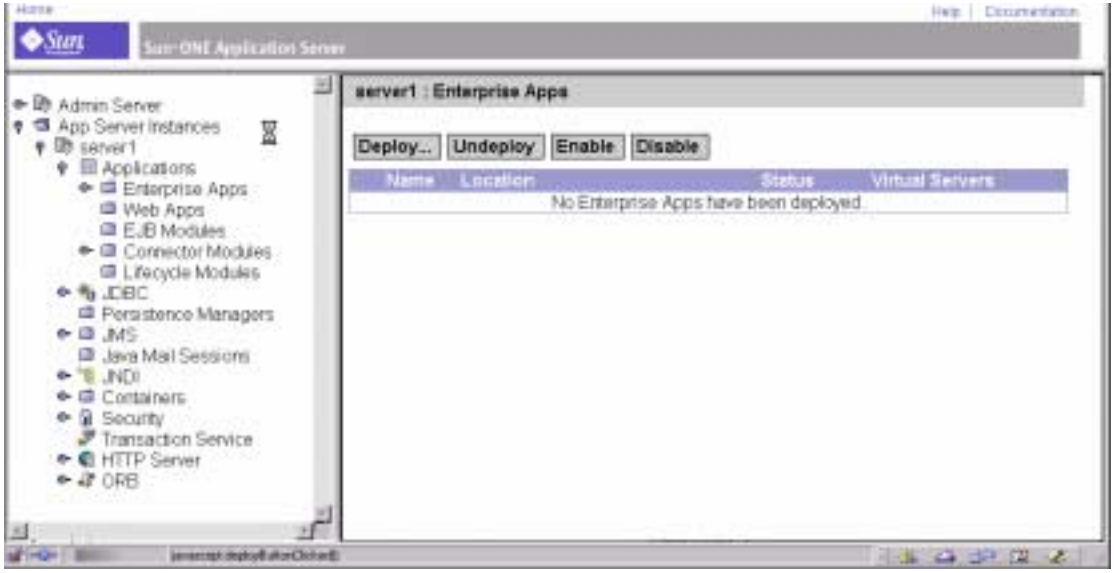
Deploying the Sample Application

The following process describes how to deploy the application.

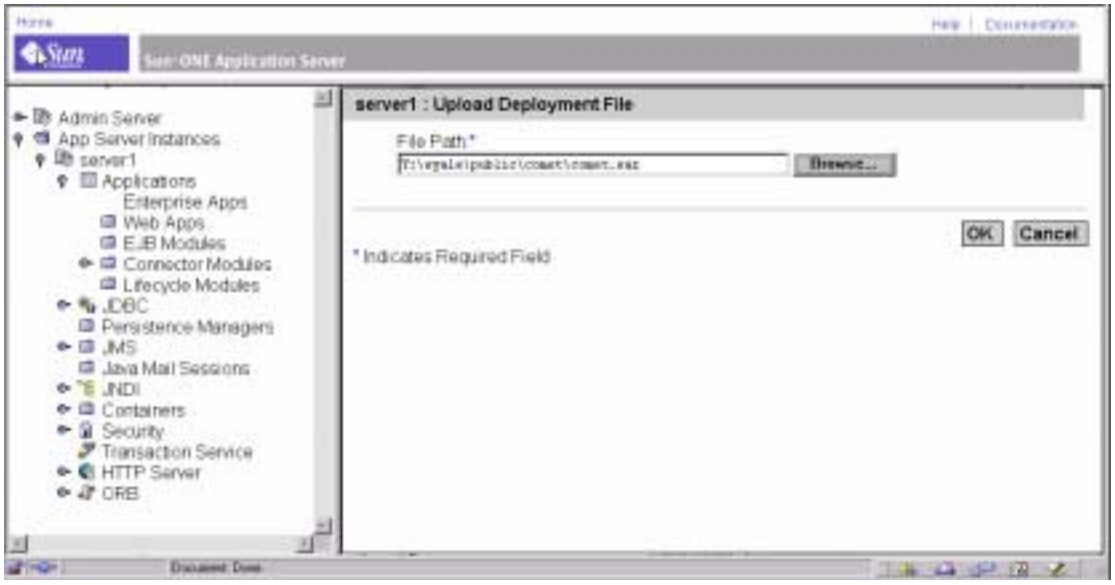
To Deploy the Sample Application

1. Load the Admin Server Tool.
2. Click the Application Server Instances-[instance name]-Applications > Enterprise Applications. See the following figure.

NOTE The default instance name is server1



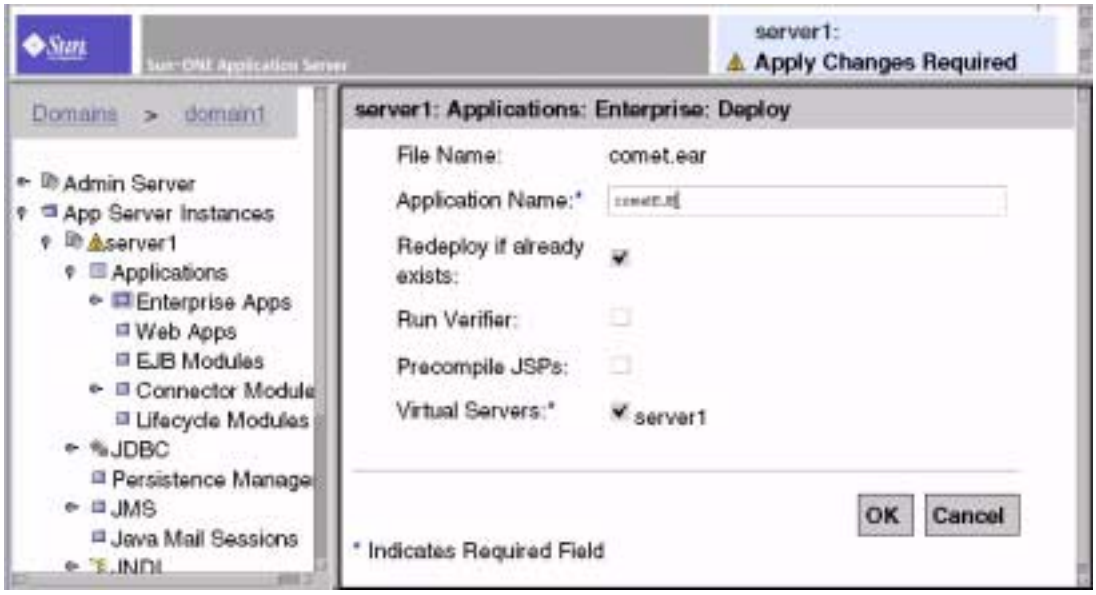
3. Click Deploy. See the following figure.



4. Browse to the `comet.ear` file path:

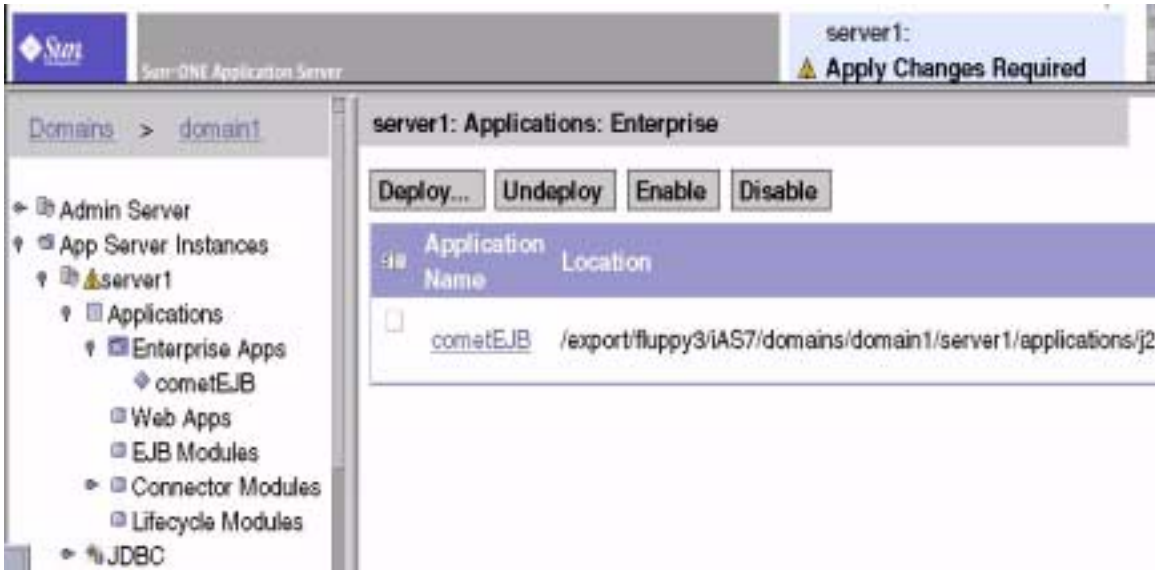
   The `comet.ear` file contains the Sample application files that are archived into .ear file.

5. Click OK. The following figure is displayed.



6. Enter the Application Name: CometEJB and click OK.

   The application is added to the Enterprise Applications list. See the following figure.



How to Operate the Sample

The following process describes how to operate the Comet sample.

To Operate the Sample

1. Execute the sample backend.

   The backend receives and sends data from sample connector using http port 8020.

2. From the `[sun7 installation directory]/sun70/samples/j2ee/backend/assemble`

   run the command: `java -classpath ./backend.jar Server 8020`

3. Load the application server.

   From the directory `[sun7 installation directory]/sun70/instances/[instance name]` run the `startserv` script.

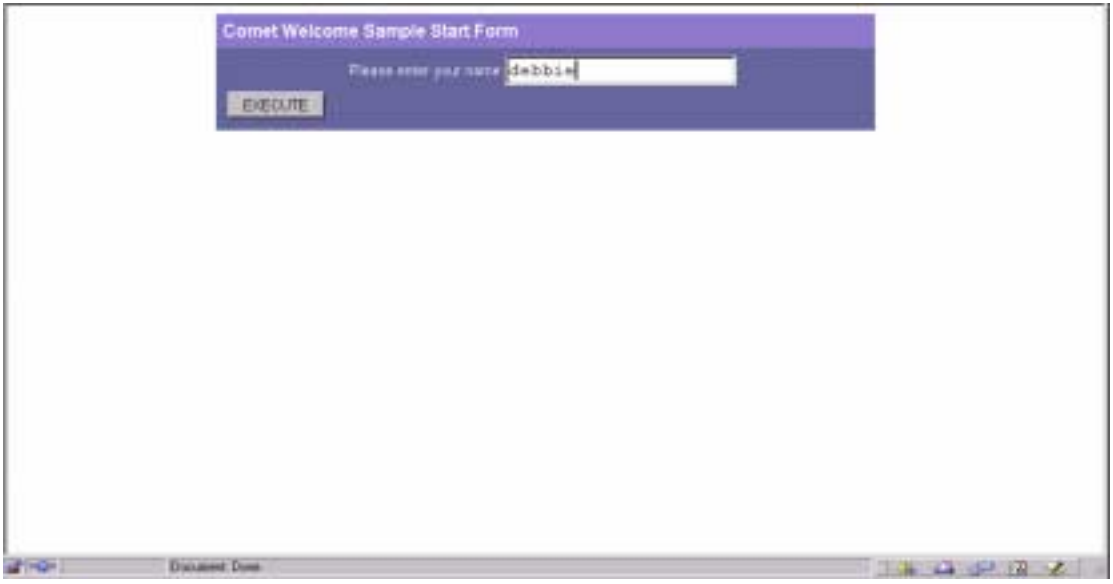
NOTE The default instance name is server1.

4. Open the browser and insert the sample url:

   ```
   http://[hostname.domainname]:[server-port]/CometEJB/startForm.jsp
   ```

NOTE The default server port is 1024.

The following figure is displayed.



5. Enter your name and click EXECUTE.

   The Hello message appears.

Appendix A

Troubleshooting

This appendix includes a description of common error messages and workarounds.

If the connector tries to create a logger an exception is thrown.
The following exception is thrown:

```
access denied(java.util.logging.LoggingPermission control)
```

Add the following line to the basic set of required permissions in the server.policy:

```
permission java.util.logging.LoggingPermission "control";
```

Glossary

This glossary provides definitions for common terms used to describe the Sun ONE Application Server deployment and development environment. For a glossary of standard J2EE terms, please see the J2EE glossary at:

```
http://java.sun.com/j2ee/glossary.html
```

access control The means of securing your Sun ONE Application Server by controlling who and what has access to it.

ACL Access Control List. ACLs are text files that contain lists identifying who can access the resources stored on your Sun ONE Application Server. *See also* general ACL.

activation The process of transferring an enterprise bean's state from secondary storage to memory.

Administration interface The set of browser based forms used to configure and administer the Sun ONE Application Server. S*ee also* CLI.

administration server An application server instance dedicated to providing the administrative functions of the Sun ONE Application Server, including deployment, browser-based administration, and access from the command-line interface (CLI) and Integrated Development Environment (IDE).

administrative domain Multiple administrative domains is a feature within the Sun ONE Application Server that allows different administrative users to create and manage their own domains. A domain is a set of instances, created using a common set of installed binaries in a single system.

API Applications Program Interface. A set of instructions that a computer program can use to communicate with other software or hardware that is designed to interpret that API.

applet A small application written in Java that runs in a web browser. Typically, applets are called by or embedded in web pages to provide special functionality. By contrast, a *servlet* is a small application that runs on a server.

application A group of components packaged into an `.ear` file with a J2EE application deployment descriptor. *See also* component, module.

application client container *See* container.

application server A reliable, secure, and scalable software platform in which business applications are run. Application servers typically provide high-level services to applications, such as component lifecycle, location, and distribution and transactional resource access,

application tier A conceptual division of a J2EE application:

client tier: The user interface (UI). End users interact with client software (such as a web browser) to use the application.

server tier: The business logic and presentation logic that make up your application, defined in the application's components.

data tier: The data access logic that enables your application to interact with a data source.

assembly The process of combining discrete components of an application into a single unit that can be deployed. S*ee also* deployment.

asynchronous communication A mode of communication in which the sender of a message need not wait for the sending method to return before it continues with other work.

attribute A name-value pair in a request object that can be set by a servlet. Also a name-value pair that modifies an element in an XML file. Contrast with *parameter*. More generally, an attribute is a unit of metadata.

auditing The method(s) by which significant events are recorded for subsequent examination, typically in error or security breach situations.

authentication The process by which an entity (such as a user) proves to another entity (such as an application) that it is acting on behalf of a specific identity (the user's security identity). Sun ONE Application Server supports basic, form-based, and SSL mutual authentication. *See also* client authentication, digest authentication, host-IP authentication, pluggable authentication.

authorization The process by which access to a method or resource is determined. Authorization in the J2EE platform depends upon whether the user associated with a request through authentication is in a given security role. For example, a human resources application may authorize managers to view personal employee information for all employees, but allow employees to only view their own personal information.

backup store A repository for data, typically a file system or database. A backup store can be monitored by a background thread (or sweeper thread) to remove unwanted entries.

bean-managed persistence Data transfer between an entity bean's variables and a data store. The data access logic is typically provided by a developer using Java Database Connectivity (JDBC) or other data access technologies. *See also* container-managed persistence.

bean-managed transaction Where transaction demarcation for an enterprise bean is controlled programmatically by the developer. *See also* container-managed transaction.

BLOB Binary Large OBject. A data type used to store and retrieve complex object fields. BLOBs are binary or serializable objects, such as pictures, that translate into large byte arrays, which are then serialized into container-managed persistence fields.

BMP *See* bean-managed persistence.

BMT *See* bean-managed transaction.

broker The Sun ONE Message Queue entity that manages JMS message routing, delivery, persistence, security, and logging, and which provides an interface that allows an administrator to monitor and tune performance and resource use.

business logic The code that implements the essential business rules of an application, rather than data integration or presentation logic.

CA See certificate authority or connector architecture.

cached rowset A `CachedRowSet` object permits you to retrieve data from a data source, then detach from the data source while you examine and modify the data. A cached row set keeps track both of the original data retrieved, and any changes made to the data by your application. If the application attempts to update the original data source, the row set is reconnected to the data source, and only those rows that have changed are merged back into the database.

Cache Control Directives Cache-control directives are a way for Sun ONE Application Server to control what information is cached by a proxy server. Using cache-control directives, you override the default caching of the proxy to protect sensitive information from being cached, and perhaps retrieved later. For these directives to work, the proxy server must comply with HTTP 1.1.

callable statement A class that encapsulates a database procedure or function call for databases that support returning result sets from stored procedures.

certificate Digital data that specifies the name of an individual, company, or other entity, and certifies that the public key included in the certificate belongs to that entity. Both clients and servers can have certificates.

certificate authority A company that sells certificates over the Internet, or a department responsible for issuing certificates for a company's intranet or extranet.

cipher A cryptographic algorithm (a mathematical function), used for encryption or decryption.

CKL Compromised Key List. A list, published by a certificate authority, that indicates any certificates that either client users or server users should no longer trust. In this case, the key has been compromised. *See also* CRL.

classloader A Java component responsible for loading Java classes according to specific rules. *See also* classpath.

classpath A path that identifies directories and JAR files where Java classes are stored. *See also* classloader.

CLI Command-line interface. An interface that enables you to type executable instructions at a user prompt. S*ee also* Administration interface.

client authentication The process of authenticating client certificates by cryptographically verifying the certificate signature and the certificate chain leading to the CA on the trust CA list. See also authentication, certificate authority.

client contract A contract that determines the communication rules between a client and the EJB container, establishes a uniform development model for applications that use enterprise beans, and guarantees greater reuse of beans by standardizing the relationship with the client.

CMP *See* container-managed persistence.

CMR *See* container-managed relationship.

CMT *See* container-managed transaction.

co-locate To position a component in the same memory space as a related component in order avoid remote procedure calls and improve performance.

column A field in a database table.

commit To complete a transaction by sending the required commands to the database. *See* rollback, transaction.

component A web application, enterprise bean, message-driven bean, application client, or connector. *See also* application, module.

component contract A contract that establishes the relationship between an enterprise bean and its container.

configuration The process of tuning the server or providing metadata for a component. Normally, the configuration for a specific component is kept in the component's deployment descriptor file. *See also* administration server, deployment descriptor.

connection factory An object that produces connection objects that enable a J2EE component to access a resource. Used to create JMS connections (TopicConnection or QueueConnection) which allow application code to make use of the provided JMS implementation. Application code uses the JNDI Service to locate connection factory objects using a JNDI Name.

Connection Pool allows highly efficient access to a database by caching and reusing physical connections, thus avoiding connection overhead and allowing a small number of connections to be shared between a large number of threads. *See also* JDBC connection pool

connector A standard extension mechanism for containers to provide connectivity to EISs. A connector is specific to an EIS and consists of a resource adapter and application development tools for EIS connectivity. The resource adapter is plugged in to a container through its support for system level contracts defined in the connector architecture.

connector architecture An architecture for the integration of J2EE applications with EISs. There are two parts to this architecture: a EIS vendor-provided resource adapter and a J2EE server that allows this resource adapter to plug in. This architecture defines a set of contracts that a resource adapter has to support to plug in to a J2EE server, for example, transactions, security and resource management.

container An entity that provides life cycle management, security, deployment, and runtime services to a specific type of J2EE component. Sun ONE Application Server provides web and EJB containers, and supports application client containers. *See also* component.

container-managed persistence Where the EJB container is responsible for entity bean persistence. Data transfer between an entity bean's variables and a data store, where the data access logic is provided by the Sun ONE Application Server. *See also* bean-managed persistence.

container-managed relationship A relationship between fields in a pair of classes where operations on one side of the relationship affect the other side.

container-managed transaction Where transaction demarcation for an enterprise bean is specified declaratively and automatically controlled by the EJB container *See also* bean-managed transaction.

control descriptor A set of enterprise bean configuration entries that enable you to specify optional individual property overrides for bean methods, plus enterprise bean transaction and security properties.

conversational state Where the state of an object changes as the result of repeated interactions with the same client. *See also* persistent state.

cookie A small collection of information that can be transmitted to a calling web browser, then retrieved on each subsequent call from that browser so the server can recognize calls from the same client. Cookies are domain-specific and can take advantage of the same web server security features as other data interchange between your application and the server.

CORBA Common Object Request Broker Architecture. A standard architecture definition for object-oriented distributed computing.

COSNaming Service An an IIOP-based naming service.

CosNaming provider To support a global JNDI name space (accessible to IIOP application clients), Sun ONE Application Server includes J2EE based CosNaming provider which supports binding of CORBA references (remote EJB references).

create method A method for customizing an enterprise bean at creation.

CRL Certificate Revocation List. A list, published by a certificate authority, that indicates any certificates that either client users or server users should no longer trust. In this case, the certificate has been revoked. *See also* CKL.

data access logic Business logic that involves interacting with a data source.

database A generic term for Relational Database Management System (RDBMS). A software package that enables the creation and manipulation of large amounts of related, organized data.

database connection A database connection is a communication link with a database or other data source. Components can create and manipulate several database connections simultaneously to access data.

data source A handle to a source of data, such as a database. Data sources are registered with the iPlanet Application Server and then retrieved programmatically in order to establish connections and interact with the data source. A data source definition specifies how to connect to the source of data.

DataSource Object A DataSource object has a set of properties that identify and describe the real world data source that it represents.

declarative security Declaring security properties in the component's configuration file and allowing the component's container (for instance, a bean's container or a servlet engine) to manage security implicitly. This type of security requires no programmatic control. Opposite of programmatic security. *See* container-managed persistence.

declarative transaction See container-managed transaction.

decryption The process of transforming encrypted information so that it is intelligible again.

delegation An object-oriented technique for using the composition of objects as an implementation strategy. One object, which is responsible for the result of an operation, delegates the implementation to another object, its delegatee. For example, a classloader often delegates the loading of some classes to its parent.

deployment The process of distributing the files required by an application to an application server to make the application available to run on the application server. *See also* assembly.

deployment descriptor An XML file provided with each module and application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve.

destination resource An objects that represents Topic or Queue destinations. Used by applications to read / write to Queues or publish / subscribe to Topics. Application code uses the JNDI Service to locate JMS resource objects using a JNDI Name.

digest authentication A for of authentication that allows the user to authenticate based on user name and password without sending the user name and password as cleartext.

digital signature an electronic security mechanism used to authenticate both a message and the signer.

directory server See Sun ONE Directory Server.

Distinguished Name *See* DN, DN attribute.

distributable session A user session that is distributable among all servers in a cluster.

distributed transaction A single transaction that can apply to multiple heterogeneous databases that may reside on separate servers.

Document Root The document root (sometimes called the primary document directory) is the central directory that contains all the virtual server's files you want to make available to remote clients.

Domain Registry The Domain Registry is a single data structure that contains domain-specific information, for all the domains created and configured on an installation of Sun ONE Application Server, such as domain name, domain location, domain port, domain host.

DN Distinguished Name. The string representation for the name of an entry in a directory server.

DN attribute Distinguished Name attribute. A text string that contains identifying information for an associated user, group, or object.

DTD Document Type Definition. A description of the structure and properties of a class of XML files.

dynamic redeployment The process of redeploying a component without restarting the server.

dynamic reloading The process of updating and reloading a component without restarting the server. By default, servlet, JavaServer Page (JSP), and enterprise bean components can be dynamically reloaded. Also known as versioning.

EAR file Enterprise ARchive file. An archive file that contains a J2EE application. EAR files have the `.ear` extension. *See also* JAR file.

e-commerce Electronic commerce. A term for business conducted over the Internet.

EIS Enterprise Information System. This can be interpreted as a packaged enterprise application, a transaction system, or a user application. Often referred to as an EIS. Examples of EISs include: R/3, PeopleSoft, Tuxedo, and CICS.

EJB container *See* container.

EJB QL EJB Query Language. A query language that provides for navigation across a network of entity beans defined by container-managed relationships.

EJB technology An enterprise bean is a server-side component that encapsulates the business logic of an application. The business logic is the code that fulfills the purpose of the application. In an inventory control application, for example, the enterprise beans might implement the business logic in methods called `checkInventoryLevel` and `orderProduct`. By invoking these methods, remote clients can access the inventory services provided by the application. *See also* container, entity bean, message-driven bean, and session bean.

ejbc utility The compiler for enterprise beans. It checks all EJB classes and interfaces for compliance with the EJB specification, and generates stubs and skeletons.

element A member of a larger set; for example, a data unit within an array, or a logic element. In an XML file, it is the basic structural unit. An XML element contains subelements or data, and may contain attributes.

encapsulate To localize knowledge within a module. Because objects encapsulate data and implementation, the user of an object can view the object as a black box that provides services. Instance variables and methods can be added, deleted, or changed, but if the services provided by the object remain the same, code that uses the object can continue to use it without being rewritten.

encryption The process of transforming information so it is unintelligible to anyone but the intended recipient.

entity bean An enterprise bean that relates to physical data, such as a row in a database. Entity beans are long lived, because they are tied to persistent data. Entity beans are always transactional and multi-user aware. *See* message-driven bean, read-only bean, session bean.

ERP Enterprise Resource Planning. A multi-module software system that supports enterprise resource planning. An ERP system typically includes a relational database and applications for managing purchasing, inventory, personnel, customer service, shipping, financial planning, and other important aspects of the business.

event A named action that triggers a response from a module or application.

external JDNI resource Allows the JNDI Service to act as a bridge to a remote JNDI server.

facade Where an application-specific stateful session bean is used to manage various Enterprise JavaBeans (EJBs).

factory class A class that creates persistence managers. *See also* connection factory.

failover A recovery process where a bean can transparently survive a server crash.

finder method Method which enables clients to look up a bean or a collection of beans in a globally available directory.

File Cache The file cache contains information about files and static file content. The file cache is turned on by default.

firewall an electronic boundary that allows a network administrator to restrict the flow of information across networks in order to enforce security.

form action handler A specially defined method in servlet or application logic that performs an action based on a named button on a form.

FQDN Fully Qualified Domain Name. The full name of a system, containing its hostname and its domain name.

general ACL A named list in the Sun ONE Directory Server that relates a user or group with one or more permissions. This list can be defined and accessed arbitrarily to record any set of permissions.

generic servlet A servlet that extends `javax.servlet.GenericServlet`. Generic servlets are protocol-independent, meaning that they contain no inherent support for HTTP or any other transport protocol. Contrast with HTTP servlet.

global database connection A database connection available to multiple components. Requires a resource manager.

global transaction A transaction that is managed and coordinated by a transaction manager and can span multiple databases and processes. The transaction manager typically uses the XA protocol to interact with the database backends. *See* local transaction.

granularity level The approach to dividing an application into pieces. A *high level of granularity* means that the application is divided into many smaller, more narrowly defined Enterprise JavaBeans (EJBs). A *low level of granularity* means the application is divided into fewer pieces, producing a larger program.

group A group of users that are related in some way. Group membership is usually maintained by a local system administrator. *See* user, role.

handle An object that identifies an enterprise bean. A client may serialize the handle, and then later deserialize it to obtain a reference to the bean.

Heuristic Decision The transactional mode used by a particular transaction. A transaction has to either Commit or Rollback.

home interface A mechanism that defines the methods that enable a client to create and remove an enterprise bean.

host-IP authentication A security mechanism used for of limiting access to the Administration Server, or the files and directories on a web site by making them available only to clients using specific computers.

HTML Hypertext Markup Language. A coding markup language used to create documents that can be displayed by web browsers. Each block of text is surrounded by codes that indicate the nature of the text.

HTML page A page coded in HTML and intended for display in a web browser.

HTTP Hypertext Transfer Protocol. The Internet protocol that fetches hypertext objects from remote hosts. It is based on TCP/IP.

HTTP servlet A servlet that extends `javax.servlet.HttpServlet`. These servlets have built-in support for the HTTP protocol. Contrast with generic servlet.

HTTPS HyperText Transmission Protocol, Secure. HTTP for secure transactions.

IDE Integrated Development Environment. Software that allows you to create, assemble, deploy, and debug code from a single, easy-to-use interface.

IIOP Internet Inter-ORB Protocol. Transport-level protocol used by both Remote Method Invocation (RMI) over IIOP and Common Object Request Broker Architecture (CORBA).

IIOP Listener The IIOP listener is a listen socket that listens on a specified port and accepts incoming connections from CORBA based client application

IMAP Internet Message Access Protocol.

IP address A structured, numeric identifier for a computer or other device on a TCP/IP network. The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Each number can be zero to 255. For example, 123.231.32.2 could be an IP address.

isolation level *See* transaction isolation level.

J2EE Java 2 Enterprise Edition. An environment for developing and deploying multi-tiered, web-based enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing these applications.

JAF The JavaBeans Activation Framework (JAF) integrates support for MIME data types into the Java platform. See Mime Types.

JAR file Java ARchive file. A file used for aggregating many files into one file. JAR files have the.`jar` extension.

JAR file contract Java ARchive contract that specifies what information must be in the enterprise bean package.

JAR file format Java ARchive file format. A platform-independent file format that aggregates many files into one file. Multiple applets and their requisite components (class files, images, sounds, and other resource files) can be bundled in a JAR file and subsequently downloaded to a browser in a single HTTP transaction. The JAR files format also supports file compression and digital signatures.

JavaBean A portable, platform-independent reusable component model.

Java IDL Java Interface Definition Language. APIs written in the Java programming language that provide a standards-based compatibility and connectivity with Common Object Request Broker Architecture (CORBA).

JavaMail session An object used by an application to interact with a mail store. Application code uses the JNDI Service to locate JavaMail session resources objects using a JNDI name.

JAXM Java API for XML Messaging. Enables applications to send and receive document-oriented XML messages using the SOAP standard. These messages can be with or without attachments.

JAXP Java API for XML Processing. A Java API that supports processing of XML documents using DOM, SAX, and XSLT. Enables applications to parse and transform XML documents independent of a particular XML processing implementation.

JAXR Java API for XML Registry. Provides a uniform and standard Java API for accessing different kinds of XML registries. Enables users to build, deploy and discover web services.

JAX-RPC Java API for XML-based Remote Procedure Calls. Enables developers to build interoperable web applications and web services based on XML-based RPC protocols.

JDBC Java Database Connectivity. A standards-based set of classes and interfaces that enable developers to create data-aware components. JDBC implements methods for connecting to and interacting with data sources in a platform- and vendor-independent way.

JDBC connection pool A pool that combines the JDBC data source properties used to specify a connection to a database with the connection pool properties.

JDBC resource A resource used to connect an application running within the application server to a database using an existing JDBC connection pool. Consists of a JNDI name (which is used by the application) and the name of an existing JDBC connection pool.

JDK Java Development Kit. The software that includes the APIs and tools that developers need to build applications for those versions of the Java platform that preceded the Java 2 Platform. *See also* JDK.

JMS Java Message Service. A standard set of interfaces and semantics that define how a JMS client accesses the facilities of a JMS message service. These interfaces provide a standard way for Java programs to create, send, receive, and read messages.

JMS-administered object A pre-configured JMS object—a connection factory or a destination—created by an administrator for use by one or more JMS clients.

The use of administered objects allows JMS clients to be provider-independent; that is, it isolates them from the proprietary aspects of a provider. These objects are placed in a JNDI name space by an administrator and are accessed by JMS clients using JNDI lookups.

JMS client An application (or software component) that interacts with other JMS clients using a JMS message service to exchange messages.

JMS connection factory The JMS administered object a JMS client uses to create a connection to a JMS message service.

JMS destination The physical destination in a JMS message service to which produced messages are delivered for routing and subsequent delivery to consumers. This physical destination is identified and encapsulated by an JMS administered object that a JMS client uses to specify the destination for which it is producing messages and / or from which it is consuming messages.

JMS messages Asynchronous requests, reports, or events that are consumed by JMS clients. A message has a header (to which additional fields can be added) and a body. The message header specifies standard fields and optional properties. The message body contains the data that is being transmitted.

JMS provider A product that implements the JMS interfaces for a messaging system and adds the administrative and control functions needed for a complete product.

JMS Service Software that provides delivery services for a JMS messaging system, including connections to JMS clients, message routing and delivery, persistence, security, and logging. The message service maintains physical destinations to which JMS clients send messages, and from which the messages are delivered to consuming clients.

JNDI Java Naming and Directory Interface. This is a standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services.

JNDI name A name used to access a resource that has been registered in the JNDI naming service.

JRE Java Runtime Environment. A subset of the Java Development Kit (JDK) consisting of the Java virtual machine, the Java core classes, and supporting files that provides runtime support for applications written in the Java programming language. *See also* JDK.

JSP JavaServer Page. A text page written using a combination of HTML or XML tags, JSP tags, and Java code. JSPs combine the layout capabilities of a standard browser page with the power of a programming language.

jspc utility The compiler for JSPs. It checks all JSPs for compliance with the JSP specification.

JTA Java Transaction API. An API that allows applications and J2EE servers to access transactions.

JTS Java Transaction Service. The Java service for processing transactions.

key-pair file *See* trust database.

LDAP Lightweight Directory Access Protocol. LDAP is an open directory access protocol that runs over TCP/IP. It is scalable to a global size and millions of entries. Using Sun ONE Directory Server, a provided LDAP server, you can store all of your enterprise's information in a single, centralized repository of directory information that any application server can access through the network.

LDIF LDAP Data Interchange Format. Format used to represent Sun ONE Directory Server entries in text form.

lifecycle event A stage in the server life cycle, such as startup or shutdown.

lifecycle module A module that listens for and performs its tasks in response to events in the server life cycle.

Listener A class, registered with a posting object, that says what to do when an event occurs.

local database connection The transaction context in a local connection is local to the current process and to the current data source, not distributed across processes or across data sources.

local interface An interface that provides a mechanism for a client that is located in the same Java Virtual Machine (JVM) with a session or entity bean to access that bean.

local session A user session that is only visible to one server.

local transaction A transaction that is native to one database and is restricted within a single process. Local transactions work only against a single backend. Local transactions are typically demarcated using JDBC APIs. *See also* global transaction.

mapping The ability to tie an object-oriented model to a relational model of data, usually the schema of a relational database. The process of converting a schema to a different structure. Also refers to the mapping of users to security roles.

MDB *See* message-driven bean.

message-driven bean An enterprise bean that is an asynchronous message consumer. A message-driven bean has no state for a specific client, but its instance variables may contain state across the handling of client messages, including an open database connection and an object reference to an EJB object. A client accesses a message-driven bean by sending messages to the destination for which the message-driven bean is a message listener.

messaging A system of asynchronous requests, reports, or events used by enterprise applications that allows loosely coupled applications to transfer information reliably and securely.

metadata Information about a component, such as its name, and specifications for its behavior.

management information base (MIB) A tree-like structure that defines the variables the master SNMP agent can access. The MIB provides access to the HTTP server's network configuration, status, and statistics. Using SNMP, you can view this information from the network management workstation (NMS). *See also* network management station (NMS) and SNMP.

MIME Data Type MIME (Multi-purpose Internet Mail Extension) types control what types of multimedia files your system supports.

module A web application, enterprise bean, message-driven bean, application client, or connector that has been deployed individually, outside an application. *See also* application, component, lifecycle module.

network management station (NMS) A machine used to remotely manage a specific network. Usually, the NMS software will provide a graph to display collected data or use that data to make sure the server is operating within a particular tolerance. *See also* SNMP.

NTV Name, Type, Value.

object persistence *See* persistence.

O/R mapping tool Object-to-relational [database] tool. A mapping tool within the Sun ONE Application Server Administrative interface that creates XML deployment descriptors for entity beans.

package A collection of related classes that are stored in a common directory. They are often literally packaged together in a Java archive JAR file. *See also* assembly, deployment.

parameter A name/value pair sent from the client, including form field data, HTTP header information, and so on, and encapsulated in a request object. Contrast with attribute. More generally, an argument to a Java method or database- prepared command.

passivation A method of releasing a bean's resources from memory without destroying the bean. In this way, a bean is made to be persistent, and can be recalled without the overhead of instantiation.

permission A set of privileges granted or denied to a user or group. *See also* ACL.

persistence For enterprise beans, the protocol for transferring the state of an entity bean between its instance variables and an underlying database. Opposite of transience. For sessions, the session storage mechanism.

persistence manager The entity responsible for the persistence of the entity beans installed in the container.

persistent state Where the state of an object is kept in persistent storage, usually a database.

pluggable authentication A mechanism that allows J2EE applications to use the Java Authentication and Authorization Service (JAAS) feature from the J2SE platform. Developers can plug in their own authentication mechanisms.

point-to-point delivery model Producers address messages to specific queues; consumers extract messages from queues established to hold their messages. A message is delivered to only one message consumer.

pooling The process of providing a number of preconfigured resources to improve performance. If a resource is pooled, a component can use an existing instance from the pool rather than instantiating a new one. In the Sun ONE Application Server, database connections, servlet instances, and enterprise bean instances can all be pooled.

POP3 Post Office Protocol

prepared command A database command (in SQL) that is precompiled to make repeated execution more efficient. Prepared commands can contain parameters. A prepared statement contains one or more prepared commands.

prepared statement A class that encapsulates a `QUERY`, `UPDATE`, or `INSERT` statement that is used repeatedly to fetch data. A prepared statement contains one or more prepared commands.

presentation layout The format of web page content.

presentation logic Activities that create a page in an application, including processing a request, generating content in response, and formatting the page for the client. Usually handled by a web application.

primary key The unique identifier that enables the client to locate a particular entity bean.

primary key class name A variable that specifies the fully qualified class name of a bean's primary key. Used for JNDI lookups.

principal The identity assigned to an entity as a result of authentication.

private key *See* public key cryptography.

process Execution sequence of an active program. A process is made up of one or more threads.

programmatic security The process of controlling security explicitly in code rather than allowing the component's container (for instance, a bean's container or a servlet engine) to handle it. Opposite of declarative security.

programmer-demarcated transaction *See* bean-managed transaction.

property A single attribute that defines the behavior of an application component. In the `server.xml` file, a property is an element that contains a name/value pair.

public key cryptography A form of cryptography in which each user has a public key and a private key. Messages are sent encrypted with the receiver's public key; the receiver decrypts them using the private key. Using this method, the private key never has to be revealed to anyone other than the user.

publish/subscribe delivery model Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to a topic. The system distributes messages arriving from a topic's multiple publishers to its multiple subscribers.

QOS QOS (Quality of Service) refers to the performance limits you set for a server instance or virtual server. For example, if you are an ISP, you might want to charge different amounts of money for virtual servers depending on how much bandwidth is provided. You can limit two areas: the amount of bandwidth and the number of connections.

queue An object created by an administrator to implement the point-to-point delivery model. A queue is always available to hold messages even when the client that consumes its messages is inactive. A queue is used as an intermediary holding place between producers and consumers.

RAR file Resource ARchive. A JAR archive that contains a resource adapter.

RDB Relational database.

RDBMS Relational database management system.

read-only bean An entity bean that is never modified by an EJB client. *See also* entity bean.

realm A scope over which a common security policy is defined and enforced by the security administrator of the security service. Also called a *security policy domain* or *security domain* in the J2EE specification.

remote interface One of two interfaces for an Enterprise JavaBean. The remote interface defines the business methods callable by a client.

request object An object that contains page and session data produced by a client, passed as an input parameter to a servlet or JavaServer Page (JSP).

resource manager An object that acts as a facilitator between a resource such as a database or message broker, and client(s) of the resource such as Sun ONE Application Server processes. Controls globally-available data sources.

resource reference An element in a deployment descriptor that identifies the component's coded name for the resource.

response object An object that references the calling client and provides methods for generating output for the client.

ResultSet An object that implements the `java.sql.ResultSet` interface. `ResultSets` are used to encapsulate a set of rows retrieved from a database or other source of tabular data.

reusable component A component created so that it can be used in more than one capacity, for instance, by more than one resource or application.

RMI Remote Method Invocation. A Java standard set of APIs that enable developers to write remote interfaces that can pass objects to remote processes.

RMIC Remote Method Invocation Compiler.

role A functional grouping of subjects in an application, represented by one or more groups in a deployed environment. *See also* user, group.

rollback Cancellation of a transaction.

row A single data record that contains values for each column in a table.

RowSet An object that encapsulates a set of rows retrieved from a database or other source of tabular data. `RowSet` extends the `java.sql.ResultSet` interface, enabling `ResultSet` to act as a JavaBeans component.

RPC Remote Procedure Call. A mechanism for accessing a remote object or service.

runtime system The software environment in which programs run. The runtime system includes all the code necessary to load programs written in the Java programming language, dynamically link native methods, manage memory, and handle exceptions. An implementation of the Java virtual machine is included, which may be a Java interpreter.

SAF Server Application Function. A function that participates in request processing and other server activities

schema The structure of the underlying database, including the names of tables, the names and types of columns, index information, and relationship (primary and foreign key) information.

Secure Socket Layer *See* SSL.

security A screening mechanism that ensures that application resources are only accessed by authorized clients.

serializable object An object that can be deconstructed and reconstructed, which enables it to be stored or distributed among multiple servers.

server instance A Sun ONE Application Server can contain multiple instances in the same installation on the same machine. Each instance has its own directory structure, configuration, and deployed applications. Each instance can also contain multiple virtual servers. *See also* virtual server.

servlet An instance of the `Servlet` class. A servlet is a reusable application that runs on a server. In the Sun ONE Application Server, a servlet acts as the central dispatcher for each interaction in an application by performing presentation logic, invoking business logic, and invoking or performing presentation layout.

servlet engine An internal object that handles all servlet metafunctions. Collectively, a set of processes that provide services for a servlet, including instantiation and execution.

servlet runner The part of the servlet engine that invokes a servlet with a request object and a response object. *See* servlet engine.

session An object used by a servlet to track a user's interaction with a web application across multiple HTTP requests.

session bean An enterprise bean that is created by a client; usually exists only for the duration of a single client-server session. A session bean performs operations for the client, such as calculations or accessing other EJBs. While a session bean may be transactional, it is not recoverable if a system crash occurs. Session bean objects can be either stateless (not associated with a particular client) or stateful (associated with a particular client), that is, they can maintain conversational state across methods and transactions. *See also* stateful session bean, stateless session bean.

session cookie A cookie that is returned to the client containing a user session identifier. *See also* sticky cookie.

session timeout A specified duration after which the Sun ONE Application Server can invalidate a user session. *See* session.

single sign-on A situation where a user's authentication state can be shared across multiple J2EE applications in a single virtual server instance.

SMTP Simple Mail Transport Protocol

SNMP SNMP (Simple Network Management Protocol) is a protocol used to exchange data about network activity. With SNMP, data travels between a managed device and a network management station (NMS). A managed device is anything that runs SNMP: hosts, routers, your web server, and other servers on your network. The NMS is a machine used to remotely manage that network.

SOAP The Simple Object Access Protocol (SOAP) uses a combination of XML-based data structuring and Hyper Text Transfer Protocol (HTTP) to define a standardized way of invoking methods in objects distributed in diverse operating environments across the Internet.

SQL Structured Query Language. A language commonly used in relational database applications. SQL2 and SQL3 designate versions of the language.

SSL Secure Sockets Layer. A protocol designed to provide secure communications on the Internet.

state **1.** The circumstances or condition of an entity at any given time. **2.** A distributed data storage mechanism which you can use to store the state of an application using the Sun ONE Application Server feature interface `IState2`. *See also* conversational state, persistent state.

stateful session bean A session bean that represents a session with a particular client and which automatically maintains state across multiple client-invoked methods.

stateless session bean A session bean that represents a stateless service. A stateless session bean is completely transient and encapsulates a temporary piece of business logic needed by a specific client for a limited time span.

sticky cookie A cookie that is returned to the client to force it to always connect to the same server process. *See also* session cookie.

stored procedure A block of statements written in SQL and stored in a database. You can use stored procedures to perform any type of database operation, such as modifying, inserting, or deleting records. The use of stored procedures improves database performance by reducing the amount of information that is sent over a network.

streaming A technique for managing how data is communicated through HTTP. When results are streamed, the first portion of the data is available for use immediately. When results are not streamed, the whole result must be received before any part of it can be used. Streaming provides a way to allow large amounts of data to be returned in a more efficient way, improving the perceived performance of the application.

Sun ONE Directory Server The Sun ONE version of Lightweight Directory Access Protocol (LDAP). Every instance of Sun ONE Application Server uses Sun ONE Directory Server to store shared server information, including information about users and groups. *See also* LDAP.

Sun ONE Message Queue The Sun ONE enterprise messaging system that implements the Java Message Service (JMS) open standard: it is a JMS provider.

system administrator The person who administers Sun ONE Application Server software and deploys Sun ONE Application Server applications.

table A named group of related data in rows and columns in a database.

thread An execution sequence inside a process. A process may allow many simultaneous threads, in which case it is multi-threaded. If a process executes each thread sequentially, it is single-threaded.

TLS Transport Layer Security. A protocol that provides encryption and certification at the transport layer, so that data can flow through a secure channel without requiring significant changes to the client and server applications.

topic An object created by an administrator to implement the publish/subscribe delivery model. A topic may be viewed as node in a content hierarchy that is responsible for gathering and distributing messages addressed to it. By using a topic as an intermediary, message publishers are kept separate from message subscribers.

transaction A set of database commands that succeed or fail as a group. All the commands involved must succeed for the entire transaction to succeed.

Transaction Attribute A transaction attribute controls the scope of a transaction.

transaction context A transaction's scope, either local or global. *See* local transaction, global transaction.

transaction isolation level Determines the extent to which concurrent transactions on a database are visible to one-another.

transaction manager An object that controls a global transaction, normally using the XA protocol. *See* global transaction.

Transaction Recovery Automatic or manual recovery of distributed transactions.

transience A protocol that releases a resource when it is not being used. Opposite of persistence.

trust database I security file that contains the public and private keys; also referred to as the key-pair file.

UDDI Universal Description, Discovery, and Integration. Provides worldwide registry of web services for discovery and integration.

URI Universal Resource Identifier. Describes a specific resource at a domain. Locally described as a subset of a base directory, so that `/ham/burger` is the base directory and a URI specifies `toppings/cheese.html`. A corresponding URL would be `http://domain:port/toppings/cheese.html`.

URL Uniform Resource Locator. An address that uniquely identifies an HTML page or other resource. A web browser uses URLs to specify which pages to display. A URL describes a transport protocol (for example, HTTP, FTP), a domain (for example, `www.my-domain.com`), and optionally a URI.

user A person who uses an application. Programmatically, a user consists of a user name, password, and set of attributes that enables an application to recognize a client. *See also* group, role.

user session A series of user application interactions that are tracked by the server. Sessions maintain user state, persistent objects, and identity authentication.

versioning *See* dynamic reloading.

virtual server A virtual web server that serves content targeted for a specific URL. Multiple virtual servers may serve content using the same or different host names, port numbers, or IP addresses. The HTTP service can direct incoming web requests to different virtual servers based on the URL. Also called a virtual host.

A web application can be assigned to a specific virtual server. A server instance can have multiple virtual servers. *See also* server instance.

WAR file Web ARchive. A Java archive that contains a web module. WAR files have the.war extension.

web application A collection of servlets, JavaServer Pages, HTML documents, and other web resources, which might include image files, compressed archives, and other data. A web application may be packaged into an archive (a WAR file) or exist in an open directory structure.

Sun ONE Application Server also supports some non-Java web application technologies, such as SHTML and CGI.

web cache An Sun ONE Application Server feature that enables a servlet or JSP to cache its results for a specific duration in order to improve performance. Subsequent calls to that servlet or JSP within the duration are given the cached results so that the servlet or JSP does not have to execute again.

web connector plug-in An extension to a web server that enables it to communicate with the Sun ONE Application Server.

web container *See* container.

web module An individually deployed web application. *See* web application.

web server A host that stores and manages HTML pages and web applications, but not full J2EE applications. The web server responds to user requests from web browsers.

Web Server Plugin The web server plugin is an HTTP reverse proxy plugin that allows you to instruct a Sun One Web Server or Sun ONE Application Server to forward certain HTTP requests to another server.

web service A service offered via the web. A self-contained, self-describing, modular application that can accept a request from a system across the Internet or an intranet, process it, and return a response.

WSDL Web Service Description Language. An XML-based language used to define web services in a standardized way. It essentially describes three fundamental properties of a web service: definition of the web service, how to access that web service, and the location of that web service.

XA protocol A database industry standard protocol for distributed transactions.

XML Extensible Markup Language. A language that uses HTML-style tags to identify the kinds of information used in documents as well as to format documents.

Index

NUMERICS

A

B

C

D

E

J

M

N

O

P

R

S

T

U

X

X